ISSN 0706-0775 X

M 2537 - N° 24 - 21 F

La Revue de l'Utilisateur de l'IBM-PC

Le magazine de l'IBM-PC et de ses compatibles

Mai 1986 - N° 24 - 21 F
Suisse 6 FS - Belgique 160 FB - Canada 3,95 $ Can.

Turbo-Pascal une version du PASCAL

Un disque virtuel grâce à Debug

POUR CHERCHER LE MEILLEUR PC VOUS POUVEZ RESTER EN EUROPE

TULIP PC extend
microprocesseur 8086, fréquence d'horloge 8 MHz. ● co processeur arithméthique (opt.) 8087/8 MHz. ● mémoire centrale (standard) 640 ko ● adaptateur graphique monochrome (standard) 80 × 25, format 720 × 348 ● adaptateur couleur graphique (opt.) 7 couleurs: 40/80 × 25, format 320/640 × 200: monochrome: 80 × 25, format 640 × 200/400 ● jeux de caractères internationaux: 11 ● interfaces: clavier compatible IBM 2; E/S parallèle: compatible Centronics; E/S série: compatible RS 232; contrôleur disques souples: 2 unités; bus d'extension pour cartes d'extension compatible IBM: 3 emplacements; piggy backed slots E/S; crayon lumineux compatible TTL ● horloge temps réel alimentée par pile ● éprom moniteur 16 ko ● unités disque souple 40 pistes 2 × 360 ko ● unité disque dur (opt.) 1 × 21 mo, 1 × 32 mo ● système d'exploitation (standard) MS-DOS 3.1 ● interpréteur (standard) GWBasic ● Window manager (standard) MS-WINDOWS avec applications MS-paint et MS-write ● unités séparées: bande magnétique, 10 mo + bande magnétique, 32 mo + bande magnétique ● set transportable ●

TULIP PC compact
microprocesseur 8088, fréquence d'horloge 8 MHz. ● coprocesseur arithmétique (opt.) 8087/8 MHz. ● mémoire centrale (standard) 512 ko ● adaptateur graphique monochrome (standard) 80 × 25, format 720 × 348 ● adaptateur graphique couleurs (opt.) 7 couleurs: 40/80 × 25, format 320/640 × 200: monochrome: 80 × 25, format 640 × 200/400 ● jeux de caractères internationaux: 11 ● interfaces: clavier compatible IBM I: E/S parallèle: compatible Centronics; E/S série: compatible RS 232; contrôleur disques souples: 2 unités; bus d'extension pour cartes d'extension compatible IBM: 4 emplacements; piggy backed slots E/S: crayon lumineux compatible TTL ● horloge temps réel ● éprom moniteur 16 ko ● unités disque souple 40 pistes 2 × 360 ko ● unité disque dur (opt.) 1 × 10 mo. ● système d'exploitation (standard) MS-DOS 3.1 ● interpréteur (standard) GWBasic ● Window manager (standard) MS-WINDOWS avec applications MS-paint et MS-write ● unités séparées: bande magnétique, 10 mo + bande magnétique ●

TULIP PC compact:
2 unités disque souple, adaptateur graphique monochrome, mémoire interne 512 ko, clavier et moniteur:
Prix public: **FFRS 14.500,–** h.t.

Tulip® computers

Siège Pays-Bas: TULIP COMPUTERS INTERNATIONAL B.V. Hambakenwetering 2 NL 5231 DC 's-Hertogenbosch. Télephone 31.73.413235, Telex 50316 nl. **Belgique:** Brusselpoortstraat 8, B 2800 Mechelen. Téléphone 32.15.422480, Télex 64698 b. **Allemagne:** Hindenburgstrasse 23, D 4300 Essen 1. Téléphone 49.201.227452.

Vers de nouvelles technologies

Daniel ARCANES

Dans ce numéro 24, de Mai 1986, Jean-Paul Pruniaux poursuit sa série sur Debug, et, commence une nouvelle étude sur le Pascal.

A l'occasion du NCC 1986, qui va se tenir à Las Vegas du 16 au 29 juin 1986, l'Ambassade des Etats-Unis à Paris avait convié la presse informatique pour une émission en directe par satellite avec Washington pour faire le point sur la technologie informatique.
Il en est ressorti que l'état du logiciel est satisfaisant et que des systèmes extrêmement complexes ont étaient mis au point.
L'attention des participants a notamment été attirée sur le fait que la productivité des programmeurs passe par une meilleur organisation des unités de travail. De façon à coordonner efficacement des activités effectuées en parallèle, et, que 1987 verrait l'apparition du Compact disque effaçable.

Les communications vont devenir, plus encore, la principale composante des PC, le lien entre les ordinateurs et les communications devenant, de plus en plus, intime.
Il sera, dès lors, aussi aisé d'appeler un ordinateur ou une base de données, où qu'ils se trouvent dans le monde, aussi simplement que d'appeler aujourd'hui un abonné de Tokyo ou de Genève.
De nouvelles technologies, rayon laser et fibres optiques notamment, occuperont le nouveau paysage informatique où le téléphone et l'ordinateur travailleront de pair.

La synthèse vocale et la reconnaissance de la parole tendront vers l'utilisation d'une communication interactive avec l'ordinateur.
Cela montre que les PC évoluent à une rapidité extrême, et, nous sommes loin du PC, tel que nous l'avions connu dans le cadre de nos premiers numéros.

Tulip® computers

COMPATIBLE ET PLUS PERFORMANT

- 8088 à 8 MHz
- Mémoire vive 256 Ko (512 Ko)
- Moniteur monochrome graphique 720×348
- Ports parallèles et série
- 2 lecteurs de disquettes 5''1/4 de 360 Ko
- MS-DOS 3-10, GW BASIC
- MS-WINDOWS Paint et Write

OPTIONS :
— Co-processeur 8087 à 8 MHz
— Version couleur : 640×200
— Disque dur 10 Mo
— Bus d'extension 4 connecteurs

13 500 F H.T.

Tulip® pc compact

PREIM'S
83, rue Michel-Ange
75016 PARIS - Tél. (1) 43.06.08.88

SOMMAIRE

4 **PASCAL**

Turbo-Pascal
une version du PASCAL
Le Pascal est devenu plus qu'un langage de programmation, il est un environnement de travail penser pour l'utilisateur.

39 **DEBUG**

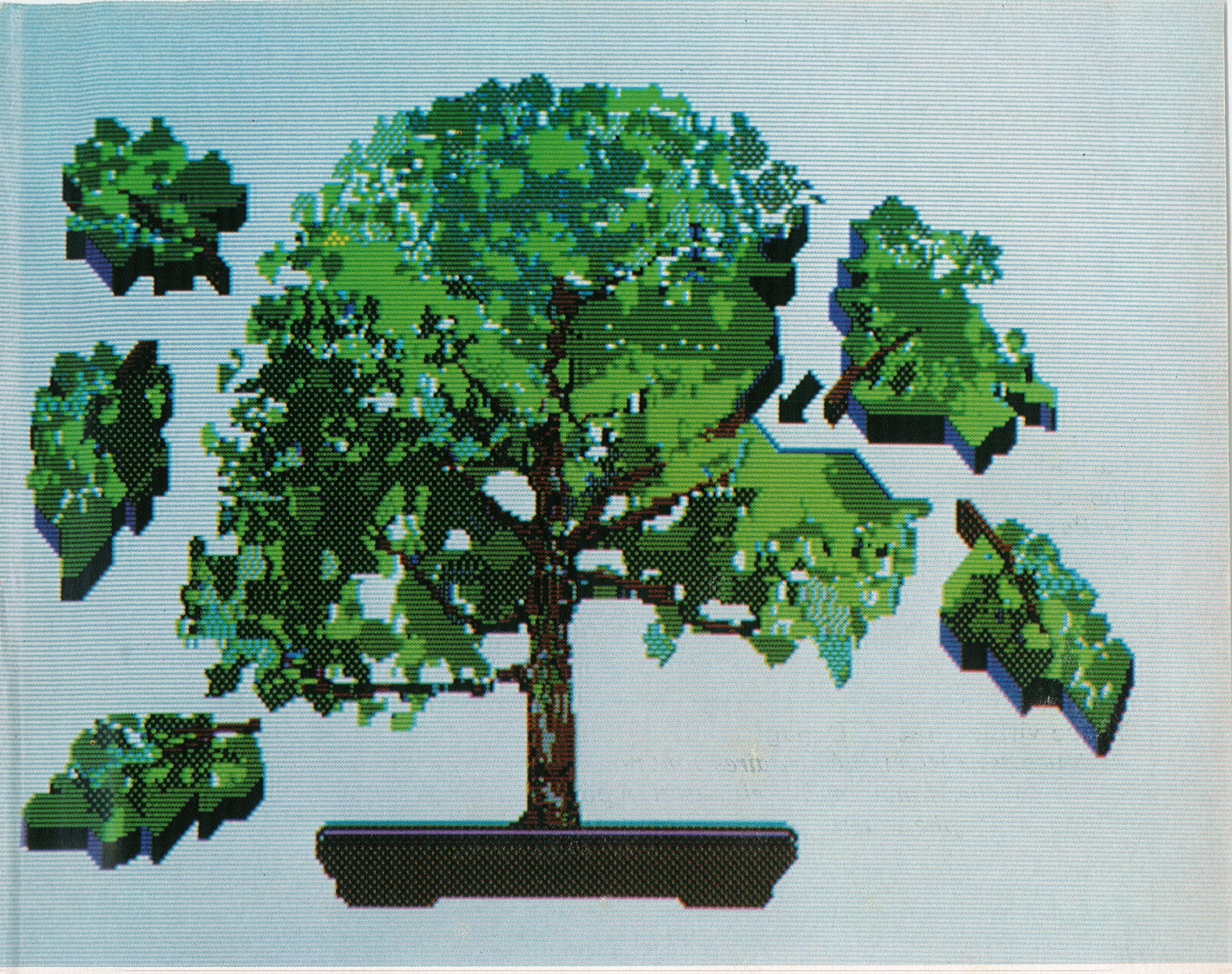
Disque virtuel grâce à Debug
Quelques données supplémentaires vont nous permettre de rajouter de nouvelles fonctions à l'initialisation pour pouvoir gérer de nouveaux périphériques.

61 **ACTUALITES**

La Revue de l'Utilisateur de l'IBM-PC
Le magazine de l'IBM-PC et de ses compatibles

Mensuel 21 F
2, rue d'Amsterdam
75009 Paris
Tél. : (1) 42.81.54.27.
Directrice de la publication : Sardie Errera
Rédacteur en chef : Roger Christophe
édité par SOFTIN COMMUNICATION (France)
Sarl R.C. Paris B 329 834 592
N° Siret 329 834 592 000 19
Commission Paritaire (en cours)
Dépôt légal : 2e trimestre 1986

CCP Paris 8850 84 D
Composition : SOFTIN COMMUNICATION
Impression : MS (Montreuil)
Les marques IBM et IBM-PC sont des marques déposées de la compagnie IBM
La société éditrice SOFTIN COMMUNICATION (France) déclare que la Revue de l'Utilisateur de l'IBM-PC n'a aucun lien avec la Compagnie IBM et que sont indépendance rédactionnelle est totale.
Accord spécial avec Publi T Service (PTS)



LA GRANDE MICRO SE FAIT MODULAIRE.

TBWA

Turbo-Pascal, une version du PASCAL

Hormis les progiciels du marché, quand nous examinons plusieurs documentations ou tarifs des éditeurs de logiciels, nous trouvons la liste des langages suivants : « Basic » et compilateur « Basic », « Fortran », « C », « Cobol », ... et à tous les coups, le « Pascal ».

Jean-Paul PRUNIAUX

Quel est donc ce langage qui semble avoir du succès aussi bien au niveau du débutant que du professionnel confirmé ?
C'est ce que nous allons chercher à savoir en étudiant une des versions de *Pascal, le Turbo-Pascal.*
A l'origine, en 1969, un professeur de Zürich, N. Wirth, avait la volonté de concevoir un langage de programmation pour donner de bonnes habitudes aux programmeurs dès le départ, ce qui en a fait un excellent outil d'enseignement de la programmation et, également, un outil efficace d'écriture d'applications complexes.

En effet, et notre expérience abonde dans le même sens, beaucoup de programmeurs ont tendance généralement à écrire n'importe quoi, n'importe comment, surtout dans les périodes de mise au point, ou, plus tard, quand quelqu'un vient les voir et leur dit : « *ce programme qui fait ceci et ceci, on ne pourrait pas aussi lui faire cela et cela en plus...* ».

A ce moment, notre programmeur s'isole dans son « *coin* », ressort ses listings, et, se pose un tas de questions : « *Bon je vais faire ceci... et puis cela... Ah zut ! je n'avait pas pensé à cela... ni à ceci... je vais donc poser une bretelle qui va m'emmener de là à là, et puis...* ».
Une bretelle, comme nul ne l'ignore dans notre jargon, est une instruction de débranchement, *GOTO* par exemple, nous faisant atterrir au beau milieu d'un autre bloc de programme.

Manque d'analyse, nous direz-vous ? Mais qui n'a jamais fait cela avec un langage comme le Basic par exemple, charger Basic et se mettre à pianoter jusqu'à ce qu'il y ait quelque chose qui en sorte.
C'est aussi quelque chose qui peut être nettement amélioré par la nature du langage, et, la première fois que nous avons eu l'occasion d'utiliser la programmation structurée, cela a été un gros progrès.

Au départ, il y a, sans conteste, une volonté d'avoir un langage clair, pratique, demandant surtout aux programmeurs de prendre de bonnes habitudes.

Le *Pascal* est basé sur les principes de la programmation structurée qui a ses règles demandant, entre autre, l'écriture en modules clairs et nets qui imposent au programmeur de mettre de l'ordre dans ses idées, et lui donne, aussi, la possibilité de rendre ses programmes parlants et lisibles. Ceci permettant ainsi au lecteur du programme, le programmeur y compris, de très vite savoir de quoi il « retourne ».

Le *Pascal,* est devenu plus qu'un langage de programmation, il est également un environnement de travail, pensé pour son utilisateur.

Prenons l'exemple du *Turbo-Pascal* qui est le logiciel avec lequel nous travaillons aujourd'hui.
Tapons la commande A×TURBO, nous voyons arriver à l'écran une première page demandant si nous désirons que *Turbo* inclu les messages d'erreur, puis une deuxième page nous donne les possibilités de travail.

Nous avons été séduit par le soin apporté à la présentation de ces pages sur l'écran ambre de notre *PC* portable, avec le texte paraissant en "*gris*" et les options à notre disposition en surbrillance.

La deuxième page d'écran est composée d'un menu suivi d'un "*prompt*" indicatif de *Turbo-Pascal,* à chaque mauvaise commande tapée, cette page nous est réaffichée.

Les options, à notre disposition, nous permettent de sélectionner le

PLUS VOUS GRANDISSEZ PLUS ELLE SE FAIT GRANDE.

Votre entreprise ne ressemble à aucune autre. Elle est unique. Ses besoins en micro-informatique le sont aussi.

Voilà pourquoi, avec le nouveau BULL Micral 60, la Grande Micro se fait modulaire.

• La micro modulaire, c'est la micro sur mesure.

Sur BULL Micral 60, la mémoire centrale de 512 Ko est extensible jusqu'à 3,6 Mo.

Une gamme de disquettes (1,2 Mo ou 360 Ko) et de disques durs (20 Mo ou 40 Mo), permet d'ajuster aujourd'hui la capacité de stockage et la puissance de BULL Micral 60 à vos besoins réels. Et d'évoluer demain tout en préservant votre investissement initial.

Les deux systèmes d'exploitation de BULL Micral 60 (Prologue et MS-DOS*) vous donnent accès à la plus large bibliothèque de logiciels et d'applications du marché.

• La micro modulaire, c'est aussi la micro multipostes.

BULL Micral 60 vous permet de passer de l'utilisation monoposte aux systèmes multipostes. Il peut aussi devenir le système fédérateur d'un réseau local de micros compatibles. Vous obtenez ainsi un meilleur partage des ressources (mémoires, télécommunications, imprimantes, etc.) et une meilleure rentabilité de vos investissements.

De plus avec le BULL Micral 60, la carte à microprocesseur** BULL CP 8 vous apporte les innovations technologiques de la télématique (accès sélectif, sécurité informatique, dossiers portables).

Avec BULL Micral 60 vous bénéficiez également du conseil, de la formation, du service après-vente et du réseau de 200 revendeurs agréés d'un grand constructeur européen d'informatique et de bureautique.

BULL. La Grande Micro.

Pour en savoir plus sur la gamme compatible BULL Micral appelez le NUMERO VERT 05.05.30.30 (APPEL GRATUIT)

* Marque déposée de Microsoft Corp. ** Brevet Innovatron.

TBWA

programme avec lequel nous désirons travailler et mettent à notre disposition un éditeur qui obéit complètement aux principes de présentation de la programmation structurée, et de faire ce qu'il faut pour que le programme puisse être exécuté, mais nous en reparlerons au prochain chapitre.

Ceci, nous change énormément des autres langages que nous avons eu l'occasion de manipuler, où nous devions, d'une part, éditer le texte d'un coté, puis appeler un compilateur et éventuellemnt un éditeur de lien ... des outils trainant de droite et de gauche sur notre système.
Ici, nous avons, au contraire, quelque chose d'harmonieux.

NATURE DU LANGAGE

A l'origine, le *Pascal* a été conçu pour être utilisé sur un système géré par un micro-processeur "*idéal*", obéissant à des instructions appelées P-CODE. Ce micro-processeur n'existant pas, il a donc été simulé.

Le compilateur traduisait les programmes sources écrit en *Pascal,* en instructions de P-CODE, puis le système, simulant ce micro-processeur, exécutait ces instructions.

Actuellement, notre *Pascal* génère directement des instructions du *8088,* et le *PC,* nous avons la possibilité de générer le programme pour exécution, soit en mémoire, soit sur le disque, dans un fichier que le *DOS* sera ultérieurement capable d'exécuter.

Hormis le gain de temps que représente la compilation directe en mémoire, *Turbo* offre un service en appréciable lors de la détection d'erreurs à la compilation. Après signalisation de l'erreur, il appelle l'éditeur, en nous positionnant le curseur à l'endroit où il ne comprend plus.

La procédure de programmation est donc la suivante :

— Analyse du problème et préparation du programme.
— Edition du programme,
— Compilation,
— En cas d'erreur de syntaxe, retour à l'éditeur,
— Exécution du programme soit depuis *Turbo,* soit depuis le *DOS,*

SYNTAXE DES INSTRUCTIONS

La présentation des instructions est libre et nous pouvons l'étaler sur plusieurs lignes, ou même mettre 15 instructions sur la même ligne, ce que nous déconseillons fortement.

La reconnaissance d'une fin d'instruction se fait par le symbole point-virgule « ; ».

Le nombre d'espaces consécutifs n'est pas significatif, et, compte comme un seul espace, de même, les retours à la ligne ne sont pas significatifs et ont la

Partez sur une meilleure dBASE
... avec DIALOGUE 2

Ce programme a été réalisé par le progiciel
Le jeu d'essai utilisé pour les tests est le suivant :
* CLIENT (NClient char 5, Nom char 10, Ville char 10, Codepostal char 5)
* FACTURE (#Nfact char 5, NClient char 5, Datef char 8, NProd char 5, Qté char 5)
* PRODUIT(#NOprod char 5, Libellé char 20, Prix char 5)
avec 3000 clients, 6000 factures et 20 produits.

Exprimez-vous plus simplement

POURQUOI FAIRE SIMPLE QUAND ON PEUT FAIRE COMPLIQUE ?

? Veuillez COMPTER les clients dont le NOM commence par "A*" qui ont commandé le produit dont le LIBELLE est "DIALOGUE" mais uniquement avec une QUANTITÉ >"70".

POURQUOI FAIRE COMPLIQUE QUAND ON PEUT FAIRE SIMPLE !

107 secondes
5 secondes

DIALOGUE
programme pour vous.

PRL RELATIONS

valeur d'espace. Notons, de plus, la possibilité que nous avons de mettre des commentaires en lieu et place de tout espace. Ces commentaires commencent par les symboles « Accolade ouverte » ou « (* », et se terminent par « Accolade fermée » ou « *) », ce qui nous amène à une grande liberté de présentation, à charge pour le programmeur de mettre ses propres règles de présentation en place, celles-ci étant, en fait, directement suggérées par la programmation structurée.

Signalons ici le couple BEGIN ... END ; délimitant un bloc d'instructions que nous pourrions assimiler à « Ouvrez la parenthèse », et « Fermez la parenthèse ». Ce groupe d'instructions, du point de vue des instructions précédentes, et suivantes, n'est considéré que comme une seule instruction.

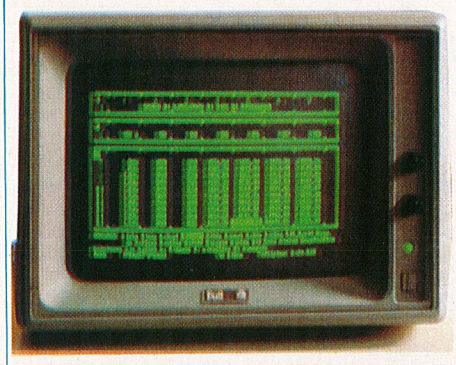
STRUCTURE D'UN PROGRAMME ET EXEMPLE

Schématiquement, un programme *Pascal* se présente de la manière suivante :

— Début de programme !
— Déclarations
— Début des instructions !
— Liste d'instructions
— Fin d'instructions ! et Fin de programme

En ce qui concerne les déclarations, le *Pascal* n'est capable de se servir d'un nom de variable ou de sous-programme, qu'une fois que vous lui avez dit ce que c'est et ceci s'appelle les déclarations.

Un nom ou identificateur peut être composé de 127 caractères, et cela va nous permettre de les choisir de manière parlante.



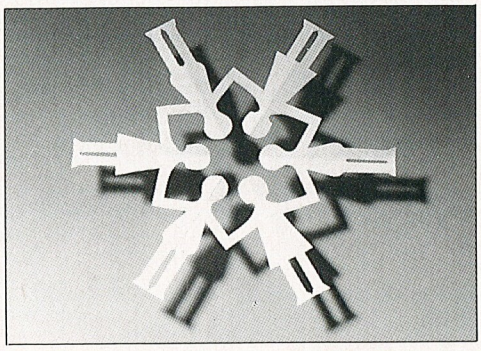
Le nom « *Annee* » parle beaucoup plus à la relecture d'un programme que « *A* ».

Voyons, tout de suite un exemple. Appelons A×TURBO, répondons Y à la question concernant les messages d'erreurs, signalons au système que nous allons travailler avec un nouveau programme que nous appelons "COUCOU", puis tapons E pour éditer ce programme dont voici le texte :

DIALOGUE 2 : BASE DE DONNÉES RELATIONNELLE

SANS PROGRAMMER

- Question/Réponse sans besoin de programmer (puissance SQL)
- Temps de réponse, en multifichiers, de l'ordre de 5 secondes
- Nombre de fichiers ouverts simultanément : 10
- Nombre de fiches par base : illimité
- Nombre de rubriques par base : 200
- Nombre de jointures actives simultanément : 200
- Nombre de liens / index par fichier / jointure : 200
- Lecture / Ecriture fichier Multiplan, Lotus 123, dBASE III, dBASE II, Wordstar, Textor, Chart, Décisionnel.
- Tous les mots clés de DIALOGUE 2 peuvent être redéfinis
- L'utilisateur crée son propre vocabulaire de l'application
- Langage d'interrogation et de mise à jour proche du français
- Dessin interactif des masques, états, lettres et étiquettes
- Restructuration automatique de la base de données
- Protection des données par mot de passe à 6 niveaux
- Partage des données en multiposte et réseau local
- Fonctionne sur IBM PC-XT-AT, BULL MICRAL 30-60, OLIVETTI M24, et compatibles. Disponible sur toute machine PROLOGUE.
- Portable :
  * PROLOGUE en multiposte, monoposte et réseau local
  * MS/DOS-PC/DOS en monoposte et réseau local 10 NET
- Prix :
  5 900 F HT : PROLOGUE et MS/DOS monoposte
  17 700 F HT : Réseau local 10 NET / DOS



POUR LES PROGRAMMEURS : LA GRANDE PRODUCTIVITÉ

DIALOGUE 2 et le langage de programmation BAL

Pour les PROGRAMMEURS, les commandes RELATIONNELLES non-procédurales de DIALOGUE 2 sont accessibles depuis le langage procédural BAL :

- Même base de données que DIALOGUE 2
- Même temps de réponse que DIALOGUE 2
- Commandes bases de données : 65
- Commandes BAL : 120
- Nombre de fichiers ouverts simultanément : 255
- Nombre de rubriques par fichier : 255
- Nombre de jointures actives simultanément : 255
- Nombre de liens / index par fichier / jointure : 255
- Nombre de variables BAL : illimité
- Nombre de segments BAL par programme : 255
- Structuré : IF-ELSE-ENDIF, WHILE-WEND, REPEAT-UNTIL
- Mise au point symbolique, Pré-processeur
- Connexion assembleur, Chargement programmes externes
- Gestion mémoire Virtuelle : mémoire illimitée
- Semi-compilé (T-code) : rapide et portable
- Extension Graphique aux normes GKS : Wiewport, Window (...)
- Extension Télécommunication : BSC 3270-2780, VIP DKU 7105, X25
- Portable : PROLOGUE en multiposte, monoposte et réseau local, MS/DOS-PC/DOS en monoposte et réseau local 10 NET
- Prix du langage BAL :
  3 500 F HT : PROLOGUE et MS/DOS monoposte
  10 500 F HT : Réseau Local 10 NET / DOS

NOTE THÉORIQUE : (pas nécessaire à l'utilisation du produit)

Une implantation originale du modèle relationnel.

DIALOGUE 2 supporte le modèle relationnel n-aire défini par E.F. CODD en 1970. Il supporte un langage relationnel du type "langage naturel". La navigation est entièrement gérée par le système. Par un mécanisme de pré-jointure, automatiquement entretenu par le système, le temps de réponse est unique. A tout moment on peut créer ou détruire une jointure à N relations. Les structures physiques sont basées sur des listes inverses multi-attributs et multirelations implantées en B-tree.
DIALOGUE 2 représente, à notre avis, le premier système réellement relationnel avec des performances meilleures que les SGBD hiérarchiques ou réseaux.
Le vocabulaire utilisé est volontairement éloigné du "jargon" relationnel.

PRL RELATIONS

prologue

H 11, AVENUE DU QUÉBEC, ZA DE COURTABOEUF, BP 73 91943 LES ULIS CÉDEX.
TÉL. (1) 69.28.85.00 / TX : 692 060 F

Codes d'accès serveur MINITEL : (1) 36.14.91.66 puis 191040521 et PRL

```
PROGRAM   Test ;            (* Debut du programme *)

VAR                         (* Declarations *)
  i : INTEGER ;

BEGIN                       (* Debut des instructions *)

  FOR   i := 1 TO 10   DO
     WRITELN  ('Coucou') ;

END.                        (* Fin des instructions *)
```

L'instruction programme débute le programme et lui donne le nom « TEST ».

Var est une instruction de déclaration, nous permettant de signaler l'utilisation de « I » comme étant un entier (Integer).

Begin commence les instructions exécutables.

For I := 1 to 10 do, est comparable à l'instruction FOR = 1 TO 10 :... :NEXT du Basic. L'équivalent du Next n'a pas lieu d'être car l'instruction qui sera reprise sera l'instruction qui suit immédiatement le DO. Si nous désirons reprendre plusieurs instructions dans cette boucle, libre à nous de faire un bloc d'instructions avec le couple BEGIN ... END ;.

Notons ici que le symbole

CONCEPTION 3D

A partir de 20 900 FHT (*) CONCEPTION 3D vous offre le logiciel le plus compétitif et le plus performant du marché.
Contactez-nous ! Notre réseau de distributeurs agréés en France et à l'étranger est à votre service.

Industries
Bureaux d'études
Architectes
Design
Schématique
Formation

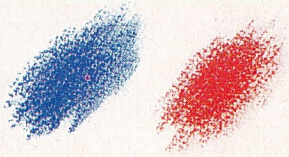
LE LEADER DE LA CAO/DAO SUR MICRO-ORDINATEURS

209/211 Boulevard Vincent Auriol 75013 PARIS
Tél.: (1) 45.86.69.00 - Télex: SERBI 206 152 F.

CONCEPTION 3D est une marque déposée de SERBI S.A.
(*) Prix au 1/5/86 comprenant les modules 3D, 2D connectable 3D et cotations.

Pierre Quinon, perchiste,
médaille d'or L.A. 1984.

Comme Pierre Quinon,
champion olympique
d'athlétisme,



l'imprimante MT 85/86
opère dans un esprit
totalement concentré
sur l'efficacité.
Elle est la plus
flexible de sa
catégorie, assurant
graphisme,
traitement de données
et traitement de textes
avec une rapidité
toute sportive.

MT 86

MANNESMANN
TALLY

8-12, av. de la Liberté - 92000 Nanterre - Tél. : (1) 47.29.14.14

Première marque d'imprimantes du marché français, sponsor de la Fédération Française d'Athlétisme, opération Euro 86.

DDRB

d'affectation traditionnellement dans les autres langages « = » (A = 10) par exemple, et signifiant « met dans ... la valeur de ... » (Met dans A la valeur 10) est ici traduit en Pascal par « := », évitant la confusion avec la signification mathématique ou logique du symbole « = ».

WriteLN demande l'impression (à l'écran) sur une ligne de ce qui suit entre parenthèse (suivie d'un passage à la ligne).

Le Begin précédent est ponctué par un End, mais comme nous sommes à la fin du programme, et que nous n'avons plus d'instructions, nous ne mettons pas le « ; ».

Le « . » suivant le End est le point final du programme.

Finissons l'édition en tapant (Ctrl-K) et (Ctrl-D), ce qui nous renvoi au menu précédent. Si vous désirez le visualiser, tapez la touche (Entrée) par exemple.

Ici, tapons C pour compiler notre programme, et, si aucune erreur n'a été faite, continuons en tapant R pour « *Run* » c'est-à-dire « *faire marcher* » notre programme.

A ce stade, nous allons voir défiler 10 messages "COUCOU" sur notre écran.

S'il y a eu erreur, le compilateur vous le signale, et vous propose de retourner sous l'éditeur, modifier ce qu'il n'a pas compris.

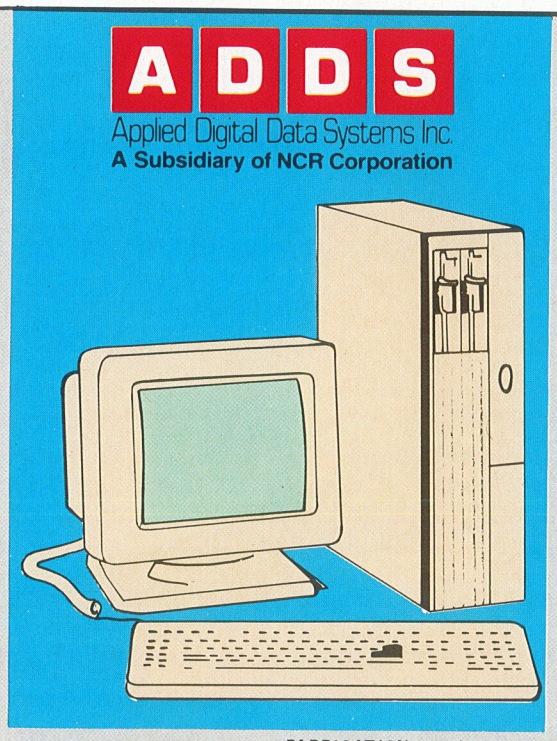
Ce qui est encore surprenant, c'est la rapidité et la souplesse à laquelle se déroulent toutes ces opérations.

DECLARATIONS DE BASE

Comme nous le disons précédemment, le *Pascal* ne peut utiliser un nom de variable, ou tout autre identificateur, qu'une fois que vous lui avez dit ce que c'est, au moyen des déclarations appropriées.

La structure d'un programme, avec ses déclarations, est a priori la suivante :

Program Nom ;
Label
— Déclarations des labels, c'est à dire de noms donnés à une instruction (Pour un GOTO par exemple) ;
Const
— Déclarations de noms désignant des valeurs constantes ;
Type
— Déclarations en plus des types existant (Entier, réel, caractère,... de nouveaux types de variables avec la signification que désire lui donner le programmeur ;
Var



FABRICATION EUROPEENNE

DÉPARTEMENT 90

LA MICRO-INFORMATIQUE

AT pour la vie

Vous êtes trop réfléchi pour vous en laisser conter par le premier compatible venu !

L'AT d'ADDS d'un prix très attractif, est un vrai 16 bits évolutif de qualité professionnelle qui vous permet de démarrer dès aujourd'hui, avec 256 Ko de RAM et d'atteindre à tous moments 4 Mo.

L'unité centrale de l'AT d' ADDS peut intégrer jusqu'à deux unités de disquettes de 360 Ko à 1,2 Mo, deux disques durs de 20 à 40 Mo et un streamer de 60 Mo.

Avec ADDS vous adapterez votre AT aux besoins de votre entreprise et vous bénéficierez de toute la puissance du fameux microprocesseur Intel 80286/8 MHz en mode monoposte sous DOS 3.1, comme en mode multiposte multitâche sous Xenix 3.1.

ATTITUDES

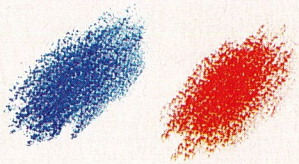
Pour plus d'informations, accompagnez votre carte de visite du coupon réponse ci-joint.

8P N° 5 - 90, rue de Villiers
92301 Levallois-Perret Cedex
Tél. : (1) 47.58.11.11 - Télex : 620 630 F
Télécopieur : 47.58.79.13

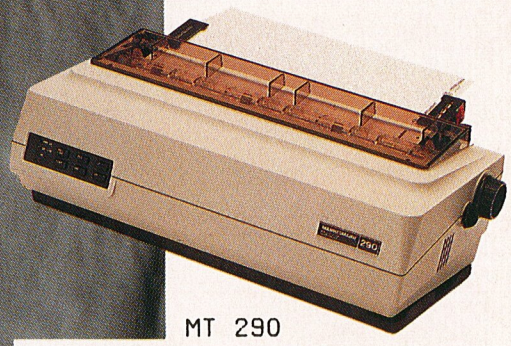
ADDS
Représenté en France par
REA
DÉPARTEMENT 90

Philippe Houvion, perchiste,
ex-recordman du monde.

Comme Philippe Houvion,
champion international
d'athlétisme,



l'imprimante MT 290
est très, très robuste.
Conçue pour mériter
la confiance
d'utilisateurs
intensifs, elle a subi
les épreuves les plus
implacables, avant
de s'imposer
comme l'outil
le plus fiable
de sa génération.



MT 290

MANNESMANN
TALLY

8-12, av. de la Liberté - 92000 Nanterre - Tél. : (1) 47.29.14.14

Première marque d'imprimantes du marché français, sponsor de la Fédération Française d'Athlétisme, opération Euro 86.

DDRB

— Déclarations des variables utilisées ;
Définitions des sous-programmes ;

Begin
— Instructions du programme principal ;

End.
Voyons un exemple qui va permettre de récapituler toutes les déclarations de base :

```
PROGRAM Declar ;                (* Exemples de declarations *)
                                (*-------------------------*)

CONST                         (* Declarations des constantes *)

     CEntier     = 10 ;           (* Constante entiere   *)
     CReel       = 10.00 ;        (* Constante reelle    *)
     CCaractere  = 'A' ;          (* Constante caractere *)
     CLogique    = TRUE ;         (* Constante logique   *)

VAR                           (* Declarations des variables *)

     VEntier     : INTEGER ;      (* Variable entiere   *)
     VReel       : REAL ;         (* Variable reelle    *)
     VCaractere  : CHAR ;         (* Variable caractere *)
     VLogique    : BOOLEAN ;      (* Variable logique   *)

BEGIN
    WRITELN ('Constantes :', 1,        1.0,        '    ',            FALSE) ;
    WRITELN ('Constantes :', CEntier, CReel, ' ', CCaractere, ' ', CLogique) ;

    VEntier     := 2 * CEntier ;
    VReel       := CReel / CEntier ;
    VCaractere  := CCaractere ;
    VLogique    := (VEntier < 12) ;

    WRITELN ('Variables  :', VEntier, VReel, ' ', VCaractere, ' ', VLogique) ;

END.
```

Ce qui à l'exécution nous donnera ceci :

```
Running
Constantes :1   1.0000000000E+00     FALSE
Constantes :10  1.0000000000E+01 A TRUE
Variables  :20  1.0000000000E+00 A FALSE
```

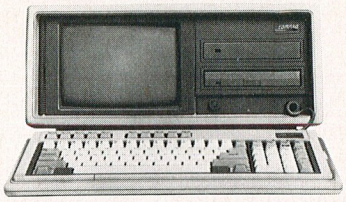
Les valeurs entières sont dans la zone classique -32 768 à 32 767.
En tant que constantes, elles peuvent aussi être exprimée en base 16 (hexadécimal) en étant précédées du symbole « $ ».

Nous pouvons aussi déclarer des entiers dans la zone 0 à 255, au moyen de la déclaration BYTE (Octet).

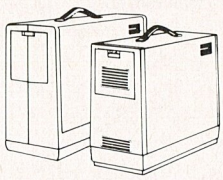
NOUVEAU. COMPAQ PORTABLE II. COMPAQ DÉFIE COMPAQ.

Ogilvy & Mather

LE DÉFI DU N° 1 MONDIAL DES PORTABLES.



Mettre au point un ordinateur portable encore plus performant, plus pratique et plus rapide que le portable le plus vendu au monde, le 1er COMPAQ Portable, c'est le défi que COMPAQ® a relevé ! Voici le COMPAQ PORTABLE II,™ un miracle pas plus gros qu'une petite valise.



30 % PLUS PETIT, IL VOUS SUIT PARTOUT.

Toutes les qualités du 1er COMPAQ Portable, et même plus : plus petit et plus léger, il vous suit de bureau en salle de réunion, chez vos clients, jusque chez vous !

COMPATIBILITÉ : IL S'ENTEND AVEC TOUT LE MONDE !

Avec sa disquette de 360 Ko, le COMPAQ PORTABLE II peut échanger des informations avec les autres ordinateurs IBM, COMPAQ et compatibles.

VITESSE : LE PUR-SANG. PUISSANCE : LE TITAN !

— 400 % plus rapide que le COMPAQ Portable, grâce au microprocesseur 80286.
— Jusqu'à 4,1 méga-octets de mémoire vive.
Du jamais vu dans un portable !

BOURRÉ DE QUALITÉS.

Taillé pour l'aventure le COMPAQ PORTABLE II : armature en aluminium renforcé, coque anti-chocs, amortisseurs pour protéger son disque dur.
Un écran haute résolution, bi-fonction : textes et graphiques.

LES AUTRES GÉANTS CHEZ VOTRE REVENDEUR AGRÉÉ COMPAQ

Avec sa gamme complète d'ordinateurs portables et de bureau, COMPAQ est aujourd'hui le n° 2 mondial des ordinateurs personnels professionnels.
Pour découvrir la gamme COMPAQ, il y a forcément un Revendeur Agréé COMPAQ près de chez vous.

COMPAQ COMPUTER France
16 (1) 64 46 36 25

COMPAQ®

LES ORDINATEURS QUI ONT EMBALLÉ L'AMÉRIQUE.

Oui, je désire recevoir plus d'informations sur COMPAQ.

RUI 1/05

Nom : _____ Fonction : _____

Société : _____

Adresse : _____

_____ Tél. _____

Retournez ce bon à
COMPAQ COMPUTER France
5, av. de Norvège BP 245 - 91944 Les Ulis Cédex

Les réels en version de base s'étalent sur
la plage 1E-38 à 1E+38, et ont une précision de 11 chiffres significatifs.

La déclaration Char, nous réserve une place mémoire pour stocker un caractère et un seul, pour les chaînes de caractères, nous allons devoir constituer des tableaux de caractères. Déclarations que nous allons apercevoir un peu plus loin. En tant que constantes, une chaîne de caractères s'exprime avec les traditionnels guillemet '...', cependant une constante caractère peut aussi être exprimée par un dièse (#) suivi du code ASCII de ce caractère, ainsi #65 sera 'A' et #27 sera la touche Escape.

Le type Boolean, nous permet de stocker une variable à 2 états, vrai ou fausse. Cette variable, à deux états, peut être obtenue à la suite d'un calcul logique, et peut être utilisée pour faire des tests. Elle peut contenir une information synthétique du style « Réponse à : Pierre est-il plus âgé que Paul ? ».

Cela représente les déclarations de variables ou constantes d'après les types de base. Maintenant, nous avons la possibilité d'ajouter de nouveaux types grâce à la déclaration Type. Nous n'en parlerons pas dans l'immédiat, et allons, tout de suite, nous intéresser aux tableaux et chaînes de caractères.

TABLEAUX ET CHAINES

Un tableau est une suite ordonnée d'informations en mémoire, auxquelles nous pouvons accèder, une à une en précisant son numéro. L'exemple que nous donnerait Basic est la déclaration DIM TAB(50).

La déclaration d'un tableau en *Pascal* se fait de la manière suivante :

(VAR ...)

— Tab : ARRAY [1..50] OF REAL ;

Celà signifie que nous déclarons Tab comme étant un tableau (array) dans lequel nous pouvons avoir accès à chaque élément, par une valeur d'indice allant de 1 à 50 ([1..50]) et dont chaque composant est un réel (OF REAL). Nous avons ainsi déclarer une liste de 50 réels.
Nous pouvons avoir accès à chacune de leur valeur par exemple par : Tab [i], où i est une valeur entre 1 et 50.

Nous pouvons aussi déclarer des tableaux à plusieurs dimensions, 2, 3,... :

Le service complet pour les applications graphiques sur IBM-PC et Compatibles

Pour les Arts graphiques, la Téléconférence, la vision industrielle, l'imagerie médicale, l'éducation et toute autre application graphique. Votre PC saisit les images par l'intermédiaire d'une caméra et d'une carte digitaliseur via un moniteur graphique, les analyse et les stocke pour obtenir les contours, le dimensionnement de pièces mécaniques, et donner les axes des trois dimensions d'une image.

24, villa de Albatros
91470 LIMOURS
Tél. : 64.91.40.40

CONSEIL - MAINTENANCE - ETUDES POUR LES APPLICATIONS GRAPHIQUES

LES TURBO
UNE SACREE FAMILLE !

Après avoir conçu TURBO-Pascal, Philippe KAHN a créé autour de ce programme toute une famille de logiciels complémentaires, qui permet de tout faire ou presque ; depuis l'enseignement sans peine du langage Pascal aux jeux, en passant par le dessin, la construction sur mesure de traitement de texte, de gestion de fichiers, etc...

TURBO Pascal - 625 F H.T. (à partir de)
Avec plus de 400.000 utilisateurs dans le monde entier, TURBO Pascal est le compilateur le plus utilisé. Vous disposez en un seul programme d'un environnement complet, éditeur et compilateur, pour programmer en Pascal. TURBO Pascal compile directement en mémoire pour plus de rapidité.

Puissance
Le langage Pascal est actuellement un des langages les plus performants sur micro-ordinateurs.
Ses applications sont nombreuses : gestion, calculs scientifiques, logiciels systèmes, graphisme, jeux, intelligence artificielle... TURBO Pascal a été retenu dans le cadre de l'opération « Informatique Pour Tous » comme support d'enseignement du langage Pascal dans les lycées et les universités. Un gage de qualité et de sérieux. Les machines 16 bits disposent de deux options : l'option 87 gère le microprocesseur 8087 pour augmenter la vitesse et la précision dans les calculs ; l'option BCD utilise la représentation décimale codée binaire pour éliminer les erreurs d'arrondi.
La version IBM comporte en plus des routines graphiques et une tortue.

Portabilité
TURBO Pascal tourne sur un grand nombre de machines, sous MS/PC-DOS, CP/M-80/86, depuis l'Amstrad jusqu'à l'IBM AT.

Prix
TURBO Pascal offre le meilleur rapport qualité/prix pour 625 F H.T. (sous CP/M-80) ou 800 F H.T. (PC/MS-DOS) vous disposez d'un éditeur plein écran et d'un compilateur Pascal complet. Ces prix comprennent le manuel de 350 pages en français. La disquette comprend de plus le code source de MicroCalc, petit tableur écrit en TURBO Pascal.

TURBO-Tutor - 350 F H.T.
Turbo-Tutor est un cours d'auto-formation à TURBO Pascal. Les débutants comme les programmeurs expérimentés y trouveront une aide précieuse dans l'écriture de leurs programmes Pascal. Ce cours comprend un manuel de 200 pages en français et une disquette avec le code source de tous les exemples.

TURBO-Graphix - 675 F H.T.
TURBO-Graphix est une librairie complète de routines graphiques haute résolution pour IBM et compatibles. Ces routines vous permettent le tracé de figures géométriques, de courbes, de polygones. Elles comprennent également tous les outils pour gérer des fenêtres. En anglais. Manuel en français disponible fin 85.

TURBO-Toolbox - 625 F H.T.
TURBO-Toolbox comprend trois utilitaires constamment utilisés par les développeurs : une gestion de fichier ISAM (par la méthode des arbres B+), une routine générale de tri et un programme générant un module d'installation pour les programmes écrits avec TURBO Pascal. Si vous développez très souvent des applications, ces outils performants vous feront gagner un temps précieux. Ils sont fournis sous forme de routines TURBO Pascal, utilisables et modifiables à volonté.

TURBO-Gameworks - 700 F H.T. NOUVEAU
Echec, Bridge et Go-Moku.
Découvrez les secrets des jeux les plus performants sur micro-ordinateurs. Ces jeux sont compilés et prêts à vous procurer de nombreuses heures de détente (ou de nuits blanches). Mais vous disposez également du code source sur votre disquette. Grâce aux sources et au manuel, la théorie des jeux n'aura plus de secrets pour vous. C'est également une façon agréable d'apprendre à programmer en Pascal. En anglais. Version française disponible début 86. Pour IBM et compatibles avec TURBO Pascal 3.0.

TURBO Editor - 700 F H.T. NOUVEAU
Construisez votre propre traitement de texte ou incorporez-le dans vos programmes. Avec TURBO Editor, vous avez : le code source prêt à être compilé, un traitement de texte complet, un manuel de 200 pages indiquant comment intégrer les procédures et les fonctions de l'éditeur dans vos programmes. TURBO Editor permet le fenêtrage. Vous pouvez ainsi éditer plusieurs documents ou plusieurs parties du même document en même temps. Pour IBM et compatibles avec TURBO Pascal 3.0.

BON A DÉCOUPER ET A RENVOYER A FRACIEL

FAMILLE TURBO PASCAL

Je désire recevoir par retour

- ☐ Turbo Pascal 3.0
  - ☐ 625 F HT pour CPM-80
  - ☐ 800 F HT pour PC/MS DOS
- ☐ Turbo 87 1.350 F HT
- ☐ Turbo Pascal BCD 1.350 F
- ☐ Turbo Pascal 87 + BCD 1.650 F HT
- ☐ Turbo-Tutor 350 F H.T.
- ☐ Turbo-Graphix 675 F H.T.
- ☐ Turbo-Toolbox 625 F H.T.
- ☐ Turbo-Gameworks 700 F H.T.
- ☐ Turbo-Editor 700 F H.T.

port 15 F TTC par produit

• TVA en sus : 18.60 %

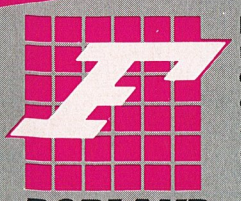
règlement joint ☐
carte bleue (date d'exp.) ___/___
contre-remboursement (+ 25 F) ☐
signature :

ordinateur : .....
Disque : ☐ 3" ☐ 3 1/2" ☐ 5 1/4" ☐ 8"
DOS : ☐ CP/M80 ☐ CP/M86 ☐ MS-DOS ☐ PC-DOS

NOM .....
ADRESSE .....
TÉL. .....

RIB

PC TOUMS 8.322.927/005 4/7 47.03.20



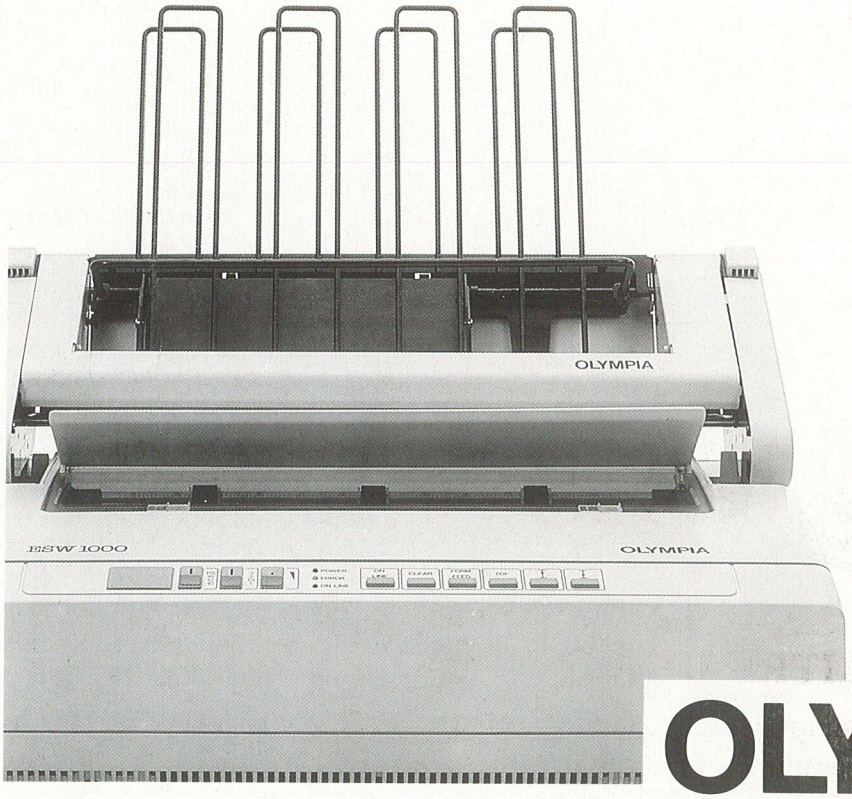
Pour vos commandes, renseignements et documentation gratuite ; ou pour contacter notre assistance téléphonique : (1) 42.72.25.19

78, rue de Turbigo
75003 Paris

IMPORTATEUR EXCLUSIF DE

Imprimantes olympiques : la qualité courrier.

Pour obtenir une frappe de qualité courrier efficace, adoptez les imprimantes à roues 100 caractères interchangeables de la gamme Olympia ESW.
Trois modèles vous sont offerts :
ESW 1000 - 20 car./sec.,
largeur papier : 375 mm,
mémoire tampon : 2 Ko
ESW 2000 - 30 car./sec.,
largeur papier : 435 mm,
mémoire tampon : 2 Ko
ESW 3000 K - 50 car./sec.,
largeur papier : 435 mm,
mémoire tampon : 4 Ko.
Les caractéristiques des Olympia ESW en font des imprimantes olympiques : échappement : 10, 12, 15 car./pouce PS. Type d'impression : caractères gras, aérés, double frappe, soulignement automatique. Impression : bidirectionnelle avec optimisation. Interface : Série V 24 - RS 232. Centronics parallèle. IEEE.
En option, vous pouvez disposer en outre d'un entraînement continu bidirectionnel, alimentation feuille à feuille 1 ou 2 bacs, clavier ESW 3000 K.
Pour vous informer, écrivez ou téléphonez à Olympia France, 10, avenue Réaumur, 92142 Clamart Cedex.
Tél. : (1) 46.30.21.42 poste 1188
Tx. : 200 771.



OLYMPIA

Performances et technique

☐ Veuillez m'adresser votre documentation gamme Olympia ESW
☐ Contactez-moi
M. .....
Raison Sociale .....
.....
Code postal ..... Ville .....
.....
Tél. .....

```
PROGRAM Tableau ;
VAR
   Tab : ARRAY [0..5, 0..5] OF INTEGER ;
   i,j : INTEGER ;

BEGIN
   FOR i := 0 TO 5 DO
      FOR j := 0 TO 5 DO
         Tab [i,j] := i+j ;

   FOR i := 0 TO 5 DO
      BEGIN
      FOR j := 0 TO 5 DO
         WRITE (' ',Tab [i,j]) ;
      WRITELN ;
      END ;

END.
```

L'exemple ci-dessus nous permet de déclarer un tableau d'entier Tab de 6 lignes et 6 colonnes numérotées de 0 à 5, ainsi que 2 entiers i et j.

Les instructions de ce programme, par une double boucle FOR remplissent ce tableau par la somme des indices.

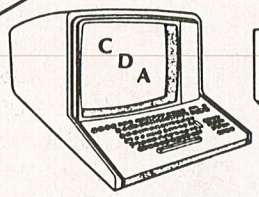
Le jeu d'instructions suivant nous permet l'impression ligne par ligne de ce tableau, par une première boucle de 0 à 5 sur i, parcourant le groupe d'instructions encadrées par BEGIN et END ;

Ce groupe d'instructions boucle entre 0 et 5 sur les valeurs j des colonnes et écrit un élément grâce à l'ordre WRITE, qui lui ne passe pas à la ligne.

Le WRITELN qui suit, assure le passage à la ligne, une fois les 6 colonnes écrites. Ceci nous donne, à l'éxécution :

```
Running
 0 1 2 3 4 5
 1 2 3 4 5 6
 2 3 4 5 6 7
 3 4 5 6 7 8
 4 5 6 7 8 9
 5 6 7 8 9 10
```


NOUVEAU



DATAMASTER

DEVENEZ LE MAITRE DE VOTRE ORDINATEUR.

— Une base de données simplifiée pour les utilisateurs avec le pouvoir de complèxité pour les programmeurs.

- Programme dirigé par menu
- Créé pour les personnes n'étant pas programmeurs
- Manuel d'utilisation très détaillé
- Recherche et analyse d'informations en français
- Trie rapide sur plusieurs clées
- Pour IBM PC et compatibles (MS - DOS 20)

Importateur Exclusif
Ste VANY International
75 bis, rue de Bellevue
92100 BOULOGNE-SUR-SEINE
Tél. (1) 46.03.75.53 - Télex 201 336

Oui, je désire recevoir la documentation **DATAMASTER.**

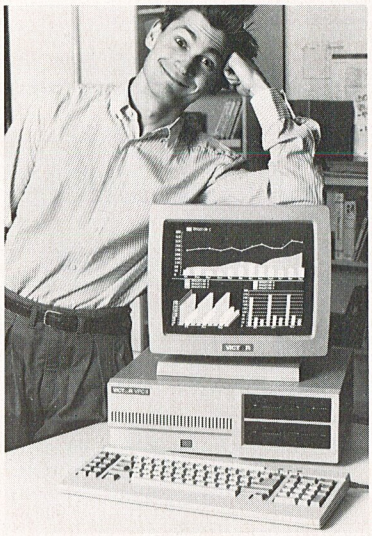
Nom ..... Société .....
Adresse .....
Code Postal ..... Tél. .....

Service lecteur n° 202 (dernière page)

Et pendant ce temps-là, Victor sort le PC2 à 11.900 F.*

(dommage pour [redacted])

Alice.



Le PC compatible que tout le monde attendait, plus performant, plus puissant et plus rapide que les PC de première génération, vient de sortir : c'est le Victor PC2.

Un compatible sous MS-DOS 3.1 qui pour seulement 11.900 F H.T. vous offre le luxe d'un vrai 16 bits grâce à son micro-processeur 8086.*** Ses 640 K RAM en standard ont la bonne puissance pour faire tourner la plupart des logiciels sans extension mémoire.**

Plus rapide que les PC de la 1re génération, en particulier le PC de [redacted], plus puissant, plus compact, il existe en deux versions : une version disque dur 20 Mo (le double du standard du marché) à 24.900 F H.T. et une version 2 disquettes 360 Ko à 11.900 F H.T.

Le PC2, il fallait qu'il existe, et il fallait être Victor pour le proposer à ce prix.

Victor Technologies. Tour Horizon, 52, quai de Dion Bouton - 92800 Puteaux. Tél. : (1) 47.78.14.50. Lyon : 72.34.12.45. Montpellier : 67.64.71.72. Nantes : 40.89.24.28.

*Prix H.T. **Marque déposée de Microsoft Corp. ***Intel Corp.

VICTOR

Comme [redacted] moins cher qu'[redacted]

Pour tous renseignements complémentaires sur le V PC 2, adressez ce coupon à Victor Technologies, Tour Horizon, 52, Quai de Dion-Bouton - 92800 Puteaux.

Nom :

Société :

Adresse :

Fonction :

Tél. :

Matériel en votre possession :

RU

Remarquez aussi que notre tableau aurait pu être déclaré :

Tab : ARRAY [0..5] OF ARRAY [0..5] OF INTEGER ;

C'est-à-dire comme une suite de liste d'entier.

De même, nous pouvons avoir accès aux éléments de ce tableau par la notation : Tab [i][j].

Les chaînes de caractères peuvent être considérées, comme des tableaux de caractères et pourraient être déclarée par : Message : ARRAY [1..32] OF CHAR ; cependant avec Turbo-Pascal nous avons une possibilité supplémentaires, qui est la déclaration STRING (Chaîne) :

Chaîne : STRING [32] ;

Cette déclaration nous permet d'utiliser des opérations sur les chaînes et nous permet de connaître le nombre de caractères utiles que possède la chaîne, grâce à la valeur contenue dans

MICRONIC C'EST
Disque dur 20 MG
+
Contrôleur Western Digital
+
Accessoires
PRIX PUBLIC : 7900F.TTC

•

MICRONIC C'EST
Extension 640 KO
Directement sur la carte mère
Pour IBM portable-COMPAQ portable
PRIX PUBLIC : 1500F.TTC

•

MICRONIC C'EST
Système MICRONIC 16pc
Compatible IBM*

•

MICRONIC C'EST
Tous les périphériques et accessoires

PRIX SPÉCIAUX POUR ÉTUDIANTS, COLLECTIVITÉS, ÉCOLES, UNIVERSITÉS.

MICRONIC

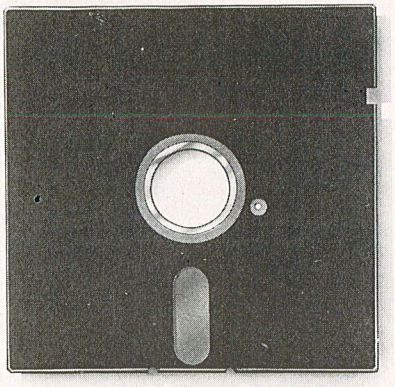
86, rue la Condamine 75017 PARIS tél. : 43.87.20.39

DISTRIBUTEURS AGRÉÉS
115, avenue d'Ivry 75013 PARIS tél. : 45.84.47.68
Provence Côte d'Azur tél. : (16) 93.69.41.80

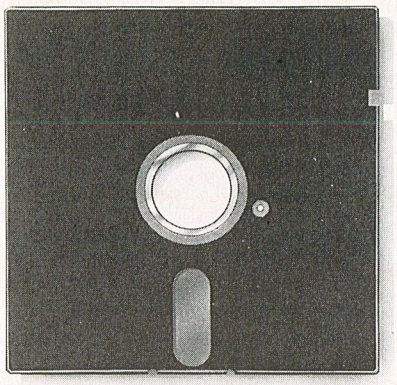
* IBM est une marque déposée

I. STUDIO (1) 42.77.06.04

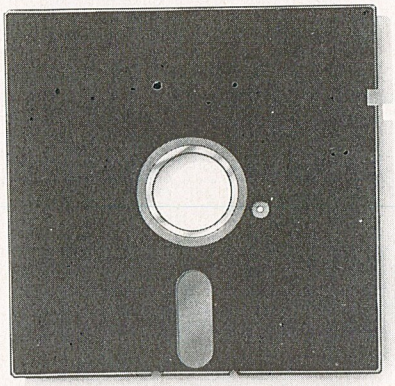
SERVICE-LECTEURS N° 170



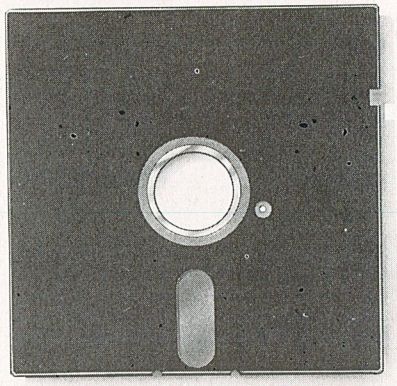
LE FABRICANT DE CETTE DISQUETTE EN A PRODUIT PLUS QUE TOUT AUTRE.



LE FABRICANT DE CETTE DISQUETTE CERTIFIE SA PRODUCTION SANS ERREUR A 100%.



LE FABRICANT DE CETTE DISQUETTE CONTRÔLE SA QUALITÉ POUR GARANTIR 30 MILLIONS DE RÉVOLUTIONS.



LE FABRICANT DE CETTE DISQUETTE. C'EST VERBATIM... LE FABRICANT DE DISQUETTES DE RENOMMÉE MONDIALE.

DE NOS JOURS, LA PROFUSION DE FOURNISSEURS BANALISE LE PRODUIT. OR, LA QUALITÉ REQUIERT UNE GRANDE EXPÉRIENCE INDUSTRIELLE.

C'EST POURQUOI 80% DES GRANDS CONSTRUCTEURS D'ORDINATEURS DU MONDE FONT CONFIANCE A VERBATIM.

C'EST POURQUOI LES UTILISATEURS SONT DE PLUS EN PLUS NOMBREUX A CHOISIR VERBATIM.

C'EST POURQUOI VERBATIM EST DEVENU LE PLUS GRAND SPÉCIALISTE AU MONDE DANS LA FABRICATION DES DISQUETTES.

FINALEMENT, ENTRE VOUS ET VERBATIM, L'ALLIANCE EST NATURELLE.

Verbatim®

LE FABRICANT DE DISQUETTES DE RENOMMÉE MONDIALE.

VERBATIM FRANCE - 33, RUE FAIDHERBE - 75011 PARIS
TÉL.: (1) 43.56.22.22 - TÉLEX: 210576 F

Chaîne [0]. Nous avons donc :

Chaîne [0] = N, nombre de caractères utiles dans la chaîne,
Chaîne [1] à Chaîne [N] = ces caractères,
Chaîne [N + 1] à ... = Caractères non utilisés.

Une remarque est à faire à propos de Chaîne [0], la déclaration de cet élément et du type CHAR, c'est-à-dire que si nous en avons besoin en tant que nombre, il faut passer par une fonction *Pascal* de conversion des variables de type non-numérique au type numérique, la fonction ORD().

Regardons, un exemple de ceci qui demande une première chaîne Cha1, la recopie dans la chaîne Cha2, affiche cette chaîne Cha2, puis l'écrit à l'écran depuis le dernier caractère saisi.

```
PROGRAM Chaine ;
VAR
   Cha1 : STRING [255] ;
   Cha2 : STRING [255] ;
   i,n  : INTEGER ;

BEGIN
   WRITELN ('Entrez des caracteres') ;
   READLN  (Cha1) ;
   Cha2 := Cha1 ;
   WRITELN (Cha2) ;

   n := ORD (Cha2[0]) ;
   FOR i := n DOWNTO 1 DO
       WRITE (Cha2 [i]) ;

END.
```

READLN () est une instruction qui permet de lire une ligne, depuis le clavier et d'affecter les variables entre parenthèses avec ce qui a été lu.

Dans n, nous mettons la longueur utile de la chaîne grâce à l'instruction ORD (), ce qui nous permet, ensuite, de faire la boucle d'affichage à l'envers (DOWNTO) sur ces n caractères. A l'exécution, nous allons avoir par exemple :

```
Running
Entrez des caracteres
A BCDE.F
A BCDE.F
F.EDCB A
```

STRUCTURES ET INSTRUCTIONS EXECUTABLES

Nous n'avons jusqu'à présent vu que les rudiments des déclarations *Pascal*, bien d'autres sont à notre disposition. Mais maintenant, regardons la partie instruction, afin de commencer à pouvoir écrire de petits programmes. Avant d'aborder les instructions structurées, examinons les opérations de base. Nous commençons par trouver des opérations arithmétiques ou logiques similaires à ce que nous pouvons trouver sous Basic, avec les symboles suivant :

+ Addition (A + B),
- Soustraction (A-B),
* Multiplication (A*B),
DIV Division entière (A Div B),
MOD Reste de la division entière (A Mod B),
/ Division
NOT Inversion logique (Not A),
AND Et logique (A And B),
OR Ou logique (A Or B),
XOR Ou exclusif (A Xor B),
=, <>, <, >, >=, <= les opérateurs relationnels (A < B)

Notons aussi la présence de 2 fonctions bien pratiques, les décalages bit à bit *SHL* et *SHR* (A Shr B) et (A Shl B) faisant défaut à plusieurs langages évolués.

Pour ceux qui ne savent pas à quoi correspond un décalage, considérez qu'il s'agit d'une multiplication entière par 2 quand il est fait à gauche, et d'une division entière par 2 quand il est fait à droite.

Nous avons aussi l'opérateur logique *IN* permettent de savoir si oui (Vrai ou True) ou non (Faux ou False) une valeur fait partie d'une liste de valeurs (A In [2,4,8,16] ou A In [0..10)]. Vous en avez un premier exemple dans le paragraphe Alternative et Tests.

Nous trouvons bien sur beaucoup d'autres fonctions du langage permettant de faire arrondis, conversions, calculs de sinus.

La règle de composition des opérations est la règle classique, comme en Basic ou Fortran, nous ne nous étendrons donc pas dessus.

Voyons maintenant les opérations structurées .

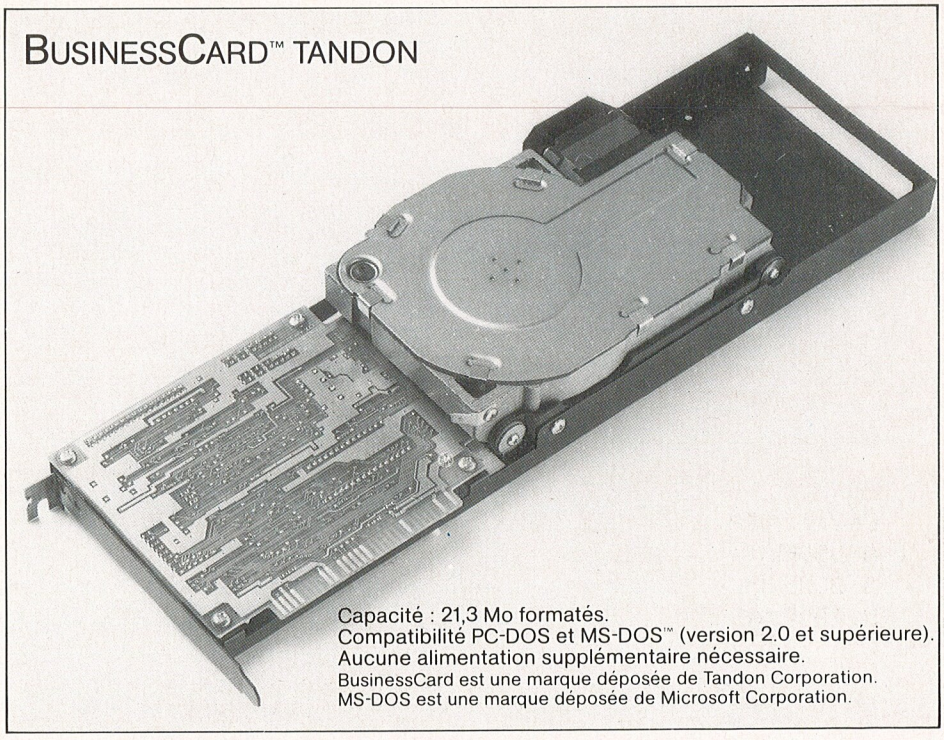
7.995 FHT*. 20 Mo. DEMANDEZ DONC A VOTRE PC CE QU'IL EN PENSE.

Dites à votre PC, s'il ne le sait pas déjà, que Tandon, le leader mondial du marché du disque magnétique, lance sa nouvelle carte : la BusinessCard.

En quelques instants et pour moins de 8.000 FHT votre PC se transforme en XT**.

Pour en savoir plus découpez donc ce bon, c'est une des rares choses qu'il ne puisse pas faire à votre place.

BUSINESSCARD™ TANDON



Capacité : 21,3 Mo formatés.
Compatibilité PC-DOS et MS-DOS™ (version 2.0 et supérieure).
Aucune alimentation supplémentaire nécessaire.
BusinessCard est une marque déposée de Tandon Corporation.
MS-DOS est une marque déposée de Microsoft Corporation.

* prix conseillé au 14/4/1986.
** XT est une marque déposée de International Business Machines Corporation.

Pour recevoir gratuitement notre documentation sur la BusinessCard, renvoyez ce coupon à : Tandon S.A. ou téléphonez au : (1) 47.60.19.00.

Nom : _____ Société : _____

Adresse : _____ Téléphone : _____

Tandon
Computer S.A.
165, boulevard de Valmy - 92706 Colombes

RU de l'IBMPC/5

BORDELAIS LEMEUNIER/LEO BURNETT

PARCOURS D'UN ENSEMBLE DE VALEUR

Nous avons déjà utilisé plusieurs fois cette instruction dans nos exemples, il s'agit de la boucle FOR. Notons cependant que celle-ci ne s'applique pas forcément à des nombres. L'instruction est de la forme suivante :

FOR (Variable) := (valeur) TO valeur DO (instruction) ;
FOR (Variable) := (valeur) DOWNTO valeur DO (instruction) ;

Ceci indique une répétition de l'instruction pour les valeurs de la variable allant de la première valeur à la dernière valeur. La première forme est à utiliser quand nous avons la valeur de départ supérieure à la valeur d'arrivée. Dans le cas contraire, nous devons utiliser la deuxième forme de cette instruction.

Les exemples de cette instruction ont tous été fait à partir de variables entières, nous pouvons en avoir avec d'autres types (mais pas de réels) :

```
PROGRAM Parcours ;
VAR
   c : CHAR ;
BEGIN
   FOR c := 'A' TO 'Z' DO
      WRITE (c) ;
   FOR c := 'z' DOWNTO 'a' DO
      WRITE (c) ;
END.
```

```
Running
ABCDEFGHIJKLMNOPQRSTUVWXYZzyxwvutsrqponmlkjihgfedcba
```

Nous avons ainsi parcouru l'ensemble des caractères compris entre 'A' et 'Z', puis celui des caractères compris entre 'z' et 'a', pour les afficher.

Afin de pouvoir faire appliquer cette instruction parcours FOR à plus d'une instruction, il n'y a pas de difficultés, regroupons ces instructions en un bloc par le couple BEGIN, END ; reprenons l'exemple ci-dessus pour, en plus, imprimer le rang du caractère :

```
PROGRAM Parcours_1 ;
VAR
   i : INTEGER ;
   c : CHAR ;
BEGIN
   i := 0 ;
   FOR c := 'A' TO 'Z' DO
      BEGIN
      i := i + 1 ;
      WRITE (c, ' ', i, ', ') ;
      END ;
FOR c := 'z' DOWNTO 'a' DO
   BEGIN
   WRITE (c, ' ', i, ', ') ;
   i := i - 1 ;
   END ;

END.
```

REPETITION TANT QUE

La deuxième opération structurée à notre disposition est la répétition Tant que (While en anglais). Sa syntaxe est la suivante :

WHILE (Condition) DO (Instruction) ;

Cette instruction assure la répétition de l'(instruction) tant que la condition est vraie. Cette condition est toujours une valeur logique Vraie ou Fausse (True ou False).

Un exemple de ceci, pourrait être l'attente de saisie de la frappe d'une touche sur le clavier par l'opérateur, information que l'on a par la fonction KEYPRESSED retournant Vrai si tel est le cas :

WHILE NOT KEYPRESSED DO ;

Ceci est la répétition de l'instruction nulle « ; » tant que pas de touches pressée sur le clavier.

```
Running
A 1, B 2, C 3, D 4, E 5, F 6, G 7, H 8, I 9, J 10, K 11, L 12, M 13, N 14, O 15,
P 16,  Q 17, R 18, S 19, T 20, U 21, V 22, W 23, X 24, Y 25, Z 26, z 26, y 25, x
24,  w 23,  v 22,  u 21, t 20, s 19, r 18, q 17, p 16, o 15, n 14, m 13, l 12, k
11, j 10, i 9, h 8, g 7, f 6, e 5, d 4, c 3, b 2, a 1,
```

Le nouveau standard statistique pour PC

Le logiciel SAS/STAT™ de SAS Institute

Offrez à votre micro-ordinateur la puissance des fonctions statistiques SAS® : avec le logiciel SAS/STAT, un logiciel intégré répondant à vos besoins statistiques sur PC.
Les fonctionnalités SAS sur PC sont les mêmes que celles sur site central ou mini-ordinateur. De plus, une interface utilisateur complètement interactive s'appuyant sur le concept de fenêtre permet d'entrer en relation avec les différentes parties de la session SAS en cours : programme, sorties, suivi de l'exécution. Et un lien micro-site central vous permet de choisir **où** et **comment** utiliser le Système SAS*.
Vous voulez gérer des données d'expérimentation de manière quotidienne ?
Faire une analyse de variance sur des plans équilibrés ou non équilibrés ?
Construire des modèles linéaires ? Le logiciel SAS/STAT répond à toutes ces demandes et à bien d'autres.

Régression/Analyse de variance

- sept types de sélection des modèles
- statistiques d'influence, diagnostics de collinéarité, courbes de régression partielles, valeurs calculées, résiduelles, et limites de confiance
- somme des carrés, statistiques de Fisher, et probalités sauvegardées dans des tableaux
- addition et suppression interactives de variables et d'observations
- possibilités d'analyse des plans à mesures répétées
- large choix de méthodes de comparaisons multiples des moyennes
- possibilité de choisir des tests d'hypothèses sur une ou plusieurs variables
- estimation de paramètres en utilisant un algorythme QR optionnel

Analyse multivariée et analyse discriminante

- en composantes principales
- analyse factorielle avec plusieurs méthodes d'estimation y compris le maximum de vraisemblance
- corrélations canoniques
- fonctions discriminantes linéaires ou quadratiques
- classification de tableaux de données en appliquant les fonctions discriminantes
- nombreuses statistiques et estimations en sortie

* Le logiciel SAS/STAT tourne sur IBM PC, PC AT, PC XT et compatibles disposant d'un disque dur et d'au moins 512 K de mémoire. L'accès aux fichiers DBF (dBase II® et dBase III®) ainsi qu'aux fichiers DIF est fourni. Pour utiliser le logiciel SAS/STAT, vous avez besoin du logiciel SAS de base pour micro-ordinateur.

* SAS est une marque déposée de SAS Institute INC., Cary, NC, USA. SAS/STAT est une marque de SAS Institute Inc. Un commentaire doit accompagner la première mention de chaque marque déposée ou de chaque marque et doit préciser que chaque marque référencée est utilisée pour identifier les produits et services de SAS Institute Inc., dBase II et dBase III sont les marques déposées d'Ashton-Tate, Culver City, CA.
Copyright© 1986 par SAS Institute

Le logiciel SAS/STAT complète le logiciel SAS de base sur micro-ordinateur. Ainsi, vous bénéficiez de la puissance du langage SAS pour vos calculs :

- création de nombres pseudo-aléatoires pour les distributions normale, binomiale, gamma, de Poisson et autres
- probabilités et quantiles pour de nombreuses distributions standard y compris les distributions de Student, Fisher et du Chi-2

De plus, le logiciel SAS de base sur PC offre un large éventail d'outils pour la manipulation des données... et des fonctionnalités statistiques simples pour les calculs de fréquence, moyennes, etc.

Un service complet sur PC

SAS Institute vous offre une gamme d'outils statistiques fiables et puissants... mais aussi une formation et de la documentation. Notre prestation comprend également le support technique, les mises à jour et une période d'essai gratuite de 45 jours.
Installez le logiciel SAS/STAT sur votre micro-ordinateur et appréciez ses capacités. Retournez le coupon-réponse ou appelez notre Département Marketing dès aujourd'hui pour plus de précisions sur nos outils statistiques sur micro-ordinateur.

SAS Institute s.a.
50, avenue Daumesnil
75579 PARIS CEDEX 12
Tél. : (1) 43 42 54 63
Télex : 216545 SASFR
Télécop. : (1) 43 47 34 64

☐ Je désire recevoir une documentation sur le logiciel SAS/STAT

☐ Je désire avoir un contact avec votre service Marketing

Veuillez compléter et nous renvoyer ce coupon

Mr Mme ..... Fonction.....

Organisme .....

Adresse .....

Code postal..... Ville .....

Téléphone .....

SAS®
SAS Institute
50, avenue Daumesnil - 75579 PARIS CEDEX 12

De la même manière que pour l'ordre FOR, nous pouvons regrouper plusieurs instructions dans un bloc BEGIN, END ; pour le faire répéter. Voyons un exemple pour afficher une série de nombres dont le suivant est obtenu en multipliant le précédent par 2, nous voulons faire ceci jusqu'à atteindre la valeur 1000.

```
PROGRAM Tant_Que ;
VAR
   i : INTEGER ;

BEGIN
   i := 1 ;
   WRITELN ('Valeur de Depart ',i) ;
   WHILE i <= 1000 DO
      BEGIN
      i := i+i ;
      WRITELN (i) ;
      END ;

END.
```

```
Running
Valeur de depart 1
2
4
8
16
32
64
128
256
512
1024
```

REPETITION JUSQU'A

Une autre instruction de répétition est :

REPEAT (Liste d'instructions) UNTIL (Condition) ;

Nous avons ici la répétition des instructions jusqu'à ce que la condition (de même nature que décrite précédemment) soit Vraie (True).

L'exemple le plus simple serait :

REPEAT UNTIL KEYPRESSED ;

C'est-à-dire répéter (l'instruction nulle) jusqu'à ce qu'une touche du clavier soit pressée.

Reprenons notre exemple précédent avec la structure REPEAT UNTIL :

```
PROGRAM Jusqu_a ;
VAR
   i : INTEGER ;

BEGIN
   i := 1 ;
   WRITELN ('Valeur de Depart ',i) ;
   REPEAT
      i := i+i ;
      WRITELN (i) ;
      UNTIL i>1000 ;

END.
```

L'informatique professionnelle pour tous avec les "PC-X" de MICROFORCE LD

UNE GAMME DU PC * A L'AT * GARANTIE UN AN DÉPANNAGE EN 72 HEURES

PC-XA 256 KO 1 LECTEUR 360 KO **6 900** F HT

PC-XB 256 KO 2 LECTEURS 360 KO **7 900** F HT*

PC-X1 640 KO 2 LECTEURS 360 KO **8 900** F HT

PC-X2 640 KO 1 LECTEUR 360 KO DISQUE DUR 20 MO **15 900** F HT

PC-ATX 6 systèmes

MICROFORCE

NOUS CONSULTER

NOS PRIX INCLUENT L'UNITÉ CENTRALE SELON DESCRIPTIF

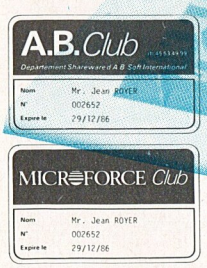
CARTE UNITÉ CENTRALE 8 SLOTS D'EXTENSION

— Processeur 8088, 4,77 MZ
— 256 KO mémoire extensible A 640 KO
— 640 KO pour PC-X1 et X21
— 1 carte multifonctions horloge, calendrier, port parallèle, port série, sortie jeux contrôleur de disquettes
— 1 carte graphique couleur et monochrome ou graphique monochrome haute résolution

Les PC-X1 et X2 sont livrés avec carte graphique multifonctions
— 1 clavier azerty compatible
— 1 système d'exploitation MS/DOS 2.11 avec licence manuel MS-DOS en français
— 1 documentation hardware complète
— 1 alimentation 135 W
— 1 boîtier métallique rigide à charnière

Tous nos systèmes sont évolutifs et peuvent supporter 4 lecteurs - Compatibilité 3ème niveau

Livraison forfaitaire 160 F HT sur toute la France.



Nous vous offrons la première année d'abonnement aux clubs

Le meilleur club de logiciels presque gratuits en prime la première disquette de logiciels

Seulement dans le cadre du club, l'accès au catalogue de produits professionnel à des prix «CLUB» software et hardware.

EXTRAIT DU CATALOGUE CLUB :

— Écran monochrome vidéo composite ..... 950 F HT
— Écran 14'' monochrome haute résolution ..... 1 700 F HT
— Écran 12'' monochrome haute résolution ..... 1 200 F HT
— Écran 14'' graphique couleur 3 fonctions : couleur, mono. vert mono. ambre ..... 3 900 F HT
— Imprimante matricielle 100 cps 80 cl ..... 2 700 F HT
— Imprimante matricielle 100 cps 132 cl ..... 3 490 F HT
— Souris PC avec carte d'adaptation ..... 990 F HT
— Disquettes DFDD 100% certifiées (10) ..... 140 F HT
— Nombreuses cartes d'extention nous consulter

N'HÉSITEZ PAS A NOUS APPELER UNE ÉQUIPE DE PROFESSIONNELS VOUS RÉPONDRA ET VOUS CONSEILLERA AVEC PLAISIR.

* Marque déposée IBM CORP.

3, rue Michel Carré
95100 ARGENTEUIL

Tél. 39.61.87.25
39.61.35.36

Télex : ATN 695 758 F

Pour plus d'informations, cerclez 172
Service lecteur télématique p. 133

Notez ici que, contrairement à l'instruction WHILE, la séquence sera toujours éxécutée au moins une fois, d'autre part, nous n'avons pas besoin du bloc BEGIN, END ; cette instruction formant déjà un bloc de par elle même.

ALTERNATIVE (TEST)

Ce que nous avons baptisé ici alternative est l'instruction de test classique Si alors ... Sinon ... qui en Pascal peut prendre les formes suivantes :

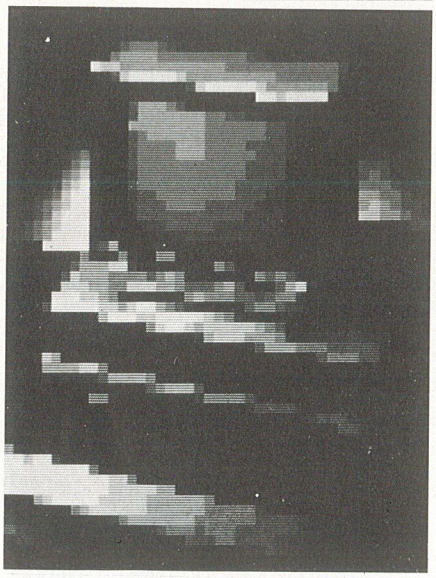
IF (Condition) THEN (Instruction) ;

IF (Condition) THEN (Instruction) ELSE (Instruction) ;

La première forme exécute l'instruction, si la condition est vraie, la deuxième forme exécute la première instruction si l'instruction, est vraie et la deuxième instruction si l'instruction est fausse, voyons en un exemple :

```
PROGRAM  Si_Alors ;
VAR
   Car : CHAR ;

BEGIN
   WRITE  ('Frappez une touche ') ;
   READLN (Car) ;

   IF  Car  IN  ['A'..'Z']  THEN
         WRITELN ('Ceci est une majuscule')
   ELSE
         WRITELN (Car,' n''est pas une majuscule') ;
END.
```

```
Running
Frappez une toucne D
Ceci est une majuscule

Running
Frappez une touche r
r n'est pas une majuscule
```



Remarquez la notation [..] qui permet de décrire un intervalle de valeur et l'utilisation de l'opérateur IN renvoyant Vrai si Car est dans cet intervalle de valeur.

Comme précédemment, nous pouvons nous servir du bloc BEGIN, END ; pour construire un groupe d'instructions :

```
IF (condition) THEN
    BEGIN
    END ;
```

CHOIX ET SELECTION

Cette strucuture très intéressante, permet d'éxécuter une instruction (ou un bloc) si le critère de choix fait partie d'une liste de valeurs :

```
CASE (paramètre) OF

(Liste 1) : (Instruction 1) ;
(Liste 2) : (Instruction 2) ;
...
ELSE
(Instruction)
END ;
```

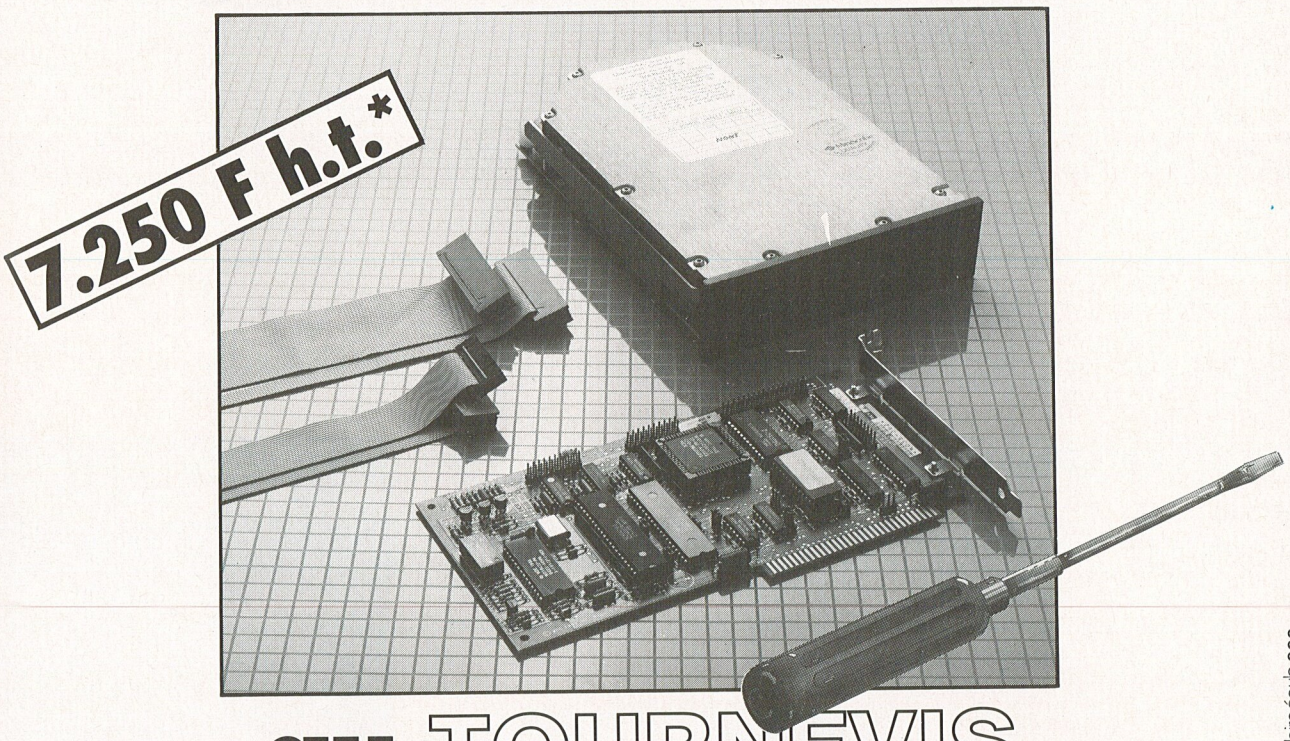
Si le paramètre fait partie de la liste 1, alors l'instruction 1 est exécutée, puis on sort de la structure Case ; sinon l'appartenance à la liste 2 est vérifiée, et ainsi de suite, jusqu'à tomber sur le END.

Le *Turbo-Pascal* améliore cette structure en rajoutant le ELSE pour le cas où aucune des listes ne conviendrait. Voyons en un exemple :

Le kit complet d'extension

QUALITÉ PROFESSIONNELLE

du WINCHESTER


7.250 F h.t.*

au TOURNEVIS

10 Mo* ou 20 Mo

DISPONIBLE SUR STOCK

* prix mars 86

birgépub 999

ERN

ERN - 237, rue Fourny - Z.A. Buc - 78530 Buc - Tél. (1) 39.56.00.11 - Télex 698 627

```
PROGRAM  Si_Alors ;
VAR
  Car : CHAR ;

BEGIN
  WRITE  ('Frappez une touche ') ;
  READLN (Car) ;

  CASE  Car  OF
    'A'..'Z' : WRITELN ('C''est une majuscule') ;
    'a'..'z' : WRITELN ('C''est une minuscule') ;
    '0'..'9' : WRITELN ('C''est un chiffre') ;
    '+','-','*','/' : WRITELN ('C''est le symbole d''une operation')
  END ;

END.
```

```
Running
Frappez une touche 4
C'est un chiffre
```

CLAVIER ET ECRAN

Avant de passer à un exemple récapitulant tout ce que nous venons de voir, attaquons, nous aux entrée/sortie avec le clavier et l'écran ; nous avons utilisé plusieurs fois dans nos exemples les instructions READ, READLN, WRITE et WRITELN, examinons plus en détail :

Ces instructions sont en fait destinées à lire, ou écrire, aussi bien dans un fichier que sur un périphérique caractère (clavier, écran, imprimante,...). Dans le cadre de fichier, pour pouvoir se servir de ces instructions, il faudra auparavant avoir déclaré et ouvert le fichier.

LECTURE

Les instructions READLN et READ obéissent à la syntaxe suivante :

READ (Liste de variables) ; ou
READ (Nom de fichier, Liste de variables) ; ou
READ (Périphérique, Liste de variables) ;

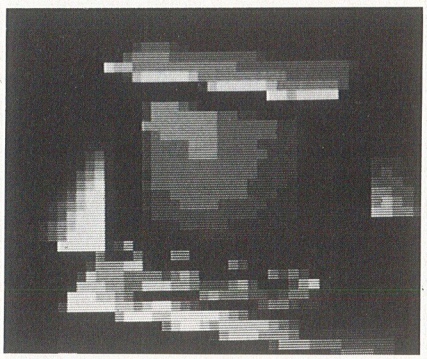
La première forme de ces instructions saisit la liste de variables depuis le clavier et en assure l'écho à l'écran, tout en permettant d'en modifier les valeurs grâce, aux touches d'édition.

La différence entre READLN et READ est l'abandon des informations en fin de ligne, avec passage à la ligne ou non.

ECRITURE

Les instructions WRITE et WRITELN obéissent à la même syntaxe que l'instruction READ (ou READLN), nous avons cependant quelques possibilités supplémentaires pour ce qui est de la présentation des sortie de type texte (pour les sorties de données sur les fichiers de données, le Pascal ne convertit à priori pas les chiffres en chaîne de chiffre, mais travaille directement avec les valeurs binaires, ce qui permet d'avoir des opérations plus rapides et des fichiers plus denses).

Ces possibilités de présentation concernent l'affichage d'une donnée sur un nombre de caractères prédéterminés, en mettant des espaces en tête de la donnée pour arriver à ce nombre. La liste de variables à écrire est de la forme :



Variable:Longueur,
Variable:Longueur, ...

Par exemple, si N vaut 12, l'instruction WRITE (N:3) ; écrira un espace, puis les chiffres 1 et 2 (soit 3 caractères en tout.

Pour les réels, l'affichage se fait avec les puissances de 10, par exemple 3.1230E+01, nous pouvons aussi le présenter en valeur décimale, grâce à la syntaxe

(Valeur:Longueur:Décimale).

Voyons un exemple de ceci :

MICRO SERVEUR VIDEOTEX ✻ MICRO SERVEUR

NOUVEAU

COHABITATION ? OUI

VTX-MAN

VTX-MAN est le seul logiciel VIDÉOTEX permettant de faire cohabiter, sur un même PC IBM, une application VIDÉOTEX avec une autre application.

VTX-MAN est un outil complet pour le développement et l'exploitation d'une application VIDÉOTEX sur Micro-ordinateur IBM PC.

CONSULTEZ-NOUS PAR MINITEL !
(16-1) 60.12.20.66

Société iSa 175, allée de la Pièce du Lavoir CHEVRY 2
91190 GIF-SUR-YVETTE - Tél.: (1) 60.12.27.54

Demande d'informations complémentaires
M.
Société :
Adresse :
Tél.

L&S

MICRO SERVEUR VIDEOTEX ✻ MICRO SERVEUR

```
PROGRAM format;

VAR
   i : INTEGER;
   r : REAL;

BEGIN
   READLN (i,r);
   WRITE  ('i =',i:6, 'r =':15, r:10, 'r =':5, r:15:7);
END.
```

```
Running
3214 7438.2539
i =  3214              r =7.4383E+03   r =    7438.2539000
```

Commentaires :
i:6 écrit 2 espaces, puis 3214,
'r =':15 écrit 12 espaces puis r, espace, et =,
r:10 écrit les 10 caractères 7.4383E+03,
'r =':5 écrit 2 espaces, puis 'r =',
r:15:7 présente les 7 chiffres après la virgule : 2539000, met devant le point, puis les chiffres 7438 et complète l'entête avec des espaces pour arriver à 15 caractères.

Ceci nous donne des possibilités de présentation très faciles à utiliser.

La différence entre WRITE et WRITELN réside dans le passage à la ligne que nous fait le WRITELN après l'écriture des données.

LES PERIPHERIQUES CARACTÈRES

Les noms Input et Output désignent les fichiers standards d'entrée-sortie (et s'appliquent en fait à l'organe d'entrée-sortie standard, le couple clavier-écran) et sont faculatatifs pour ces instructions d'entrée-sortie ; ainsi les exepressions suivantes sont équivalentes :

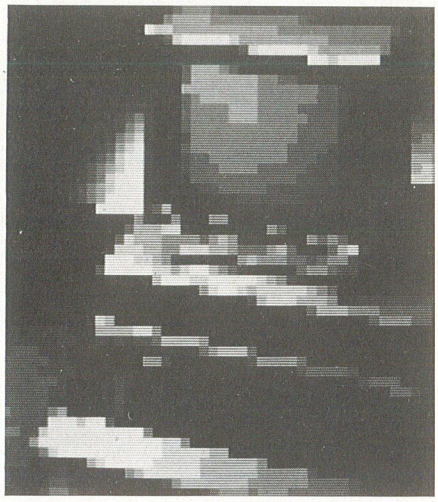
READLN (i,r) ; ou READLN (Input,i,r) ;
WRITE (i,r) ; ou WRITE (Output,i,r) ;

Nous pouvons aussi spécifier (sans autres déclarations) les noms suivants :

Con : correspondant à ce que le DOS reconnaît comme étant CON:, c'est-à-dire la console (équivalent dans l'ensemble à Input ou Output).
Kbd : Le clavier, les caractères sont lus immédiatement et sans écho.
Lst : L'imprimante (que le PC DOS reconnaît comme étant PRN:).
Aux : Le canal auxiliaire de communication (La liaison série AUX:).
Usr : Appel de fonctions d'entrée-sortie définies par l'utilisateur (par défaut il s'agit des fonctions standard du DOS, mais peuvent être redéfinies).

Ainsi la ligne suivante nous assurera, similairement à INKEY$ en Basica la saisie d'un caractère sans son écho, mais attendra la frappe de ce caractère Ch (déclaré VAR Ch : CHAR) :

READ (Kbd, Ch) ;



UN PROGRAMME

Un programme qui nous apparaît excellent, quand nous abordons un nouveau langage, est le Master-Mind. Le Master-Mind est un jeu relativement simple à programmer, mais qui nous fait passer en revue plusieurs possibilités de boucles et d'entrée-sortie.

Rappelons les règles de ce jeu. A partir d'un tas de pions de plusieurs couleurs (6 ou 7) un des joueurs choisit une chaîne mystère (1 blanc, 2 bleus, 1 rouge et 1 jaune). L'autre joueur doit trouver cette série, et pour cela propose une chaîne de pions. Le premier lui indique le nombre de pions qu'il a trouvé au bon endroit, de la bonne couleur, et le nombre de pion de la bonne couleur, mais mal placés.

Nous allons donc programmer un Master-Mind, où les couleurs vont en fait être remplacées par des lettres (A,B,C,...) que notre *IBM-PC* va tirer au hasard.

Voyons ce dont nous avons besoin. Tout d'abord, nous avons décidé de pouvoir modifier la difficulté du jeu, en modifiant le nombre de couleur ou symboles (A,B,C..) avec lesquels jouer (variable Nb—Symbol), et en modifiant la longueur de la chaîne mystère (de 2 à 10). Ainsi, les lettres faisant partie de la solution devront être comprises entre les valeurs 'A' et Max—Maj (à calculer) ou bien en minuscule 'a' et Max—Min.

Ceci nous amène à l'entête de programme suivant :

```
PROGRAM Master ;                    (* MASTER-MIND *)
                                    (*-------------*)
VAR
   A_Trouver : STRING [10] ;        (* Chaine mystere     *)
   Propose   : STRING [10] ;        (* Solution proposee *)
   Nb_Symbol,                       (* Nbre de symboles possibles differents    *)
   Longueur  : INTEGER ;            (* Longueur de la chaine mystere            *)
   Max_Maj   ,                      (* Caracteres jouables de 'A' a Max_Maj     *)
   Max_Min   : CHAR ;               (* Equivalent en minuscule de 'a' a Max_Min *)

   Reponse   : CHAR ;
   Nb_Solut,                        (* Nombre d'essais faits par le joueur      *)
   Bien_Mis,                        (* Nombre de bons symboles a la bonne place *)
   Bon_Symb  : INTEGER ;            (* Nombre de bons symboles trouves          *)
   i, j      : INTEGER ;            (* Compteur banal                           *)

   Fin_Jeu   : BOOLEAN ;            (* Vrai si le jeu en cours est fini         *)
   f         : BOOLEAN ;            (* Variable de travail                      *)
```

Notre programme va comprendre une grande boucle extérieure permettant au joueur de faire plusieurs jeux à suivre et d'en modifier la difficulté, après cette saisie des paramètres, s'ils sont bons, nous rentrons dans un jeu :

```
BEGIN
REPEAT
   CLRSCR; (* Efface l'ecran *)
   WRITELN ('Master-Mind :');
   WRITELN;

   WRITE   ('Nombre de signes utilisables   (2 a 26) : ');
   READLN  (Nb_Symbol);
   WRITE   ('Longueur de la chaine cherchee (2 a 10) : ');
   READLN  (Longueur);
   WRITELN;

   WRITELN;
   WRITE   ('Voulez-vous arreter ? ');
   READ    (KBD, Propose [1]);
   UNTIL   (Propose[1] IN ['O','o']);

END.
```

Reprenons le bloc IF (Nb—Symbol ... vérifiant que le nombre de symboles est compris entre 2 et 26 et la longueur entre 2 et 10, nous allons trouver ici un jeu, pour cela nous allons l'initialiser en tirant au sort des caractères de la chaîne mystère, auparavant nous ajustons la longueur des chaînes A - Trouver et Propose à la longueur de travail, la fonction CHR, équivalente à CHR$ du Basic nous crée un caractère à partir de son code ASCII. Randomize initialise le générateur de nombre alélatoire, Random(N) nous génère un nombre aléatoire entre 0 et N. Nb—Solut est le nombre d'essai fait par le joueur que nous initialisons à 1.

```
IF (Nb_Symbol IN [2..26]) AND (Longueur IN [2..10]) THEN
   BEGIN

   A_Trouver [0] := CHR (Longueur);
   Propose   [0] := CHR (Longueur);
   RANDOMIZE;
   FOR i := 1 TO Longueur DO
      A_Trouver [i] := CHR (65+RANDOM(Nb_Symbol));
   Max_Maj  := CHR (65+Nb_Symbol);
   Max_Min  := CHR (97+Nb_Symbol);
   Nb_Solut := 1;

   REPEAT

      (* Ici se deroule la succession de coups joues
      jusqu'a ce que le joueur gagne ou abandonne *)

      UNTIL Fin_Jeu;

   END;
```

Dans la boucle Repeat... Until, le joueur va proposer une solution, puis s'il n'a pas abandonné, le programme va évaluer la réponse. Voyons d'abord la saisie de la solution.

Après un affichage destiné à savoir où le joueur en est, nous metrons dans une boucle de saisie des caractères directement à partir du clavier pour le ranger en position i dans la chaîne A—Trouver. Si le caractère est Escape (#27), le joueur abandonne et nous affichons la solution, si c'est Effacement arrière (# 8) nous reculons d'un caractère. Si c'est une majuscule valide, nous l'affichions, la stockons et passons au caractère suivant, si c'est une minuscule valide, nous la transformaons en majuscule par la fonction Pascal UpCase () ; avant de la stocker. Puis nous rentrons dans l'évaluation de la solution si le joueur n'a pas abandonné.

POUR LA REALISATION DE TOUS VOS TRAVAUX D'IMPRESSION (journaux d'entreprises, manuels, catalogues de stages) ET VOS PLAQUETTES PUBLICITAIRES à des prix compétitifs

OP FORMAT
7, rue Jules Bourdais
75017 Paris
Tél.42.67.93.80

```
REPEAT

   WRITE (Nb_Solut:3, ' : ');
   i := 1; Fin_Jeu := False;
   WHILE (i <= Longueur) AND NOT Fin_Jeu
      BEGIN
      READ (Kbd, Reponse);
      CASE Reponse OF
#27   :  BEGIN
         Fin_Jeu := TRUE;
         WRITELN ('  Vous abandonnez, la solution est ', A_Trouver)
         END;
#8    :  IF i>1 THEN BEGIN
            WRITE (Reponse);
            i := i-1
            END;

ELSE     IF (Reponse >= 'A') AND (Reponse < Max_Maj) THEN
            BEGIN
            WRITE (Reponse);
            Propose [i] := Reponse;
            i := i+1
            END
         ELSE IF (Reponse >= 'a') AND (Reponse < Max_Min) THEN
            BEGIN
            WRITE  (Reponse);
            Propose [i] := UPCASE (Reponse);
            i := i+1
            END;

         END; {Case}
      END; {While}

      IF NOT Fin_Jeu THEN
         BEGIN

         (* Ici la solution est evaluee *)

         END ;
      UNTIL Fin_Jeu;
```

Si nous passons le cap du test If Not Fin—Jeu, nous avons là une solution du à évaluer, la première chose que nous pouvons faire est de vérifier si le joueur n'aurait pas trouvé, auquel cas, le jeu va se terminer ici ; pour cela nous pouvons réaffecter le contenu de Fin - Jeu avec le résultat de l'opération logique Propose = A - trouver qui sera vrai si le jouer a trouvé.

Si le joueur n'a pas trouvé, comptons le nombre de symboles qu'il aurait mis au bon endroit dans la variable Bien - Mais, puis nous trouverons un autre bloc que nous éxaminerons plus loin pour savoir combien le joueur a utilisé de symboles corrects et affichons les résultats :

```
IF NOT Fin_Jeu THEN
   BEGIN
   Fin_Jeu := Propose = A_Trouver;
   IF Fin_Jeu THEN
      WRITELN ('  Bravo, vous avez trouve')

   ELSE
      BEGIN
      Bien_Mis := 0;
      FOR i:= 1 TO Longueur DO
      IF Propose [i] = A_Trouver [i] THEN
         Bien_Mis := Bien_Mis + 1;

      Bon_Symb := 0;

      (* Ici est la boucle d'evaluation de
      bons symboles mis n'importe ou     *)

      WRITELN (Bon_Symb:3, ':', Bien_Mis);
      Nb_Solut := Nb_Solut + 1
      END;
   END;
```

Le dernier bloc à écrire est la recherche des bons symboles utilisés. Pour cela une boucle extérieure va nous permettre de parcourrir tous les symboles de la chaîne mystère sur l'indice i.

Nous allons aborder une deuxième boucle avec l'indice j sur les caractères de la solution proposée et récupérerons dans f la valeur vrai si un caractère de la propostion est égal à un caractère de la solution.

Nous allons ainsi parcourrir tous les caractères de la proposition pour chaque caractère mystère jusqu'à en trouver un qui corresponde, nous le comptons et le détruisons par un '*' pour éviter qu'il ne soit pris en compte par les boucles futures.

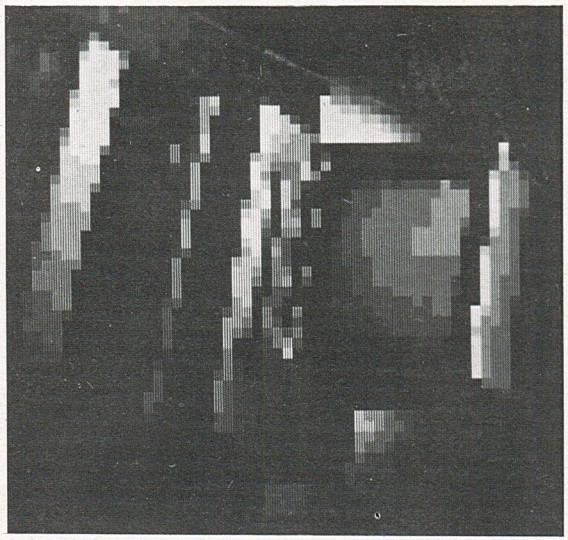
Une fois tout ce texte de programme saisi (vous en trouverez un listing complet en fin d'article), il ne nous reste plus qu'à le compiler, éventuellement de générer un fichier MASTER.COM, et à l'éxécution, nous trouvons quelque chose du genre :

```
FOR i := 1 TO Longueur DO
   BEGIN
   j := 1;
   REPEAT
      f := Propose [j] = A_Trouver [i];
      IF f THEN
         BEGIN
         Bon_Symb := Bon_Symb + 1;
         Propose [j] := '*'
         END;
      j := j + 1;
      UNTIL (j > Longueur) OR f ;
   END;
```

```
Master-Mind :

Nombre de signes utilisables   (2 a 26) : 6
Longueur de la chaine cherchee (2 a 10) : 4

  1 : ABCD  3:0
  2 : CDEF  3:0
  3 : DCAF  3:3
  4 : DCAE  Bravo, vous avez trouve

Voulez-vous arreter ?
>
```

Nous venons ici de faire un premier tour et de nous familiariser avec le Pascal, entre autres avec la version au *Pascal*.
Ceci nous a permis d'en apprendre beaucoup sur ce sujet... que nous approfondirons le mois prochain.
Turbo-Pascal, qui nous offre bien des possibilités ou extensions

```
PROGRAM Master ;                    (* MASTER-MIND *)
                                    (*-------------*)
VAR
   A_Trouver : STRING [10] ;   (* Chaine mystere    *)
   Propose   : STRING [10] ;   (* Solution proposee *)
   Nb_Symbol,                  (* Nbre de symboles possibles differents    *)
   Longueur  : INTEGER ;       (* Longueur de la chaine mystere            *)
   Max_Maj   ,                 (* Caracteres jouables de 'A' a Max_Maj     *)
   Max_Min   : CHAR ;          (* Equivalent en minuscule de 'a' a Max_Min *)

   Reponse   : CHAR ;
   Nb_Solut,                   (* Nombre d'essais faits par le joueur      *)
   Bien_Mis,                   (* Nombre de bons symboles a la bonne place *)
   Bon_Symb  : INTEGER ;       (* Nombre de bons symboles trouves          *)
   i, j      : INTEGER ;       (* Compteur banal                           *)

   Fin_Jeu   : BOOLEAN ;       (* Vrai si le jeu en cours est fini         *)
   f         : BOOLEAN ;       (* Variable de travail                      *)

BEGIN

(*----------------------------*)
(* Boucle sur plusieurs jeux *)
(*----------------------------*)

REPEAT
   CLRSCR;
   WRITELN ('Master-Mind :');
   WRITELN;

(* Saisie des parametres *)
(*-----------------------*)

   WRITE   ('Nombre de signes utilisables   (2 a 26) : ');
   READLN  (Nb_Symbol);
   WRITE   ('Longueur de la chaine cherchee (2 a 10) : ');
   READLN  (Longueur);
   WRITELN;

   IF (Nb_Symbol IN [2..26]) AND (Longueur IN [2..10]) THEN

(* Parametres valides *)
(*--------------------*)                    (* Creation aleatoire     *)
                                             (* de la chaine mystere   *)
                                             (*------------------------*)
      BEGIN
      A_Trouver [0] := CHR (Longueur);       (* Longueurs de chaine    *)
      Propose   [0] := CHR (Longueur);
      RANDOMIZE;                             (* Init. nombre aleat.    *)
      FOR i := 1 TO Longueur DO              (* Caracteres aleatoires  *)
         A_Trouver [i] := CHR (65+RANDOM(Nb_Symbol));
      Max_Maj  := CHR (65+Nb_Symbol);        (* CHR (65) = 'A'         *)
      Max_Min  := CHR (97+Nb_Symbol);        (* CHR (97) = 'a'         *)
      Nb_Solut := 1;                         (* premier coup du jeu    *)

      REPEAT

(* saisie de solutions  *)
(*----------------------*)

         WRITE (Nb_Solut:3, ' : ');                  (* Nombre de coups      *)

         i := 1; Fin_Jeu := False;                   (* Saisie de Propose    *)
         WHILE (i <= Longueur) AND NOT Fin_Jeu       (*----------------------*)
            BEGIN
            READ (Kbd, Reponse);                       (* Lecture d'un car.  *)

            CASE Reponse OF
#27         :  BEGIN                                       (* Escape :      *)
               Fin_Jeu := TRUE;
               WRITELN ('  Vous abandonnez, la solution est ', A_Trouver)
               END;

#8          :  IF i>1 THEN BEGIN                           (* Effacement :  *)
                  WRITE (Reponse);
                  i := i-1
                  END;

ELSE        IF (Reponse >= 'A') AND (Reponse < Max_Maj) THEN
               BEGIN                                      (* Majuscule     *)
               WRITE (Reponse);
               Propose [i] := Reponse;
               i := i+1
               END

            ELSE
            IF (Reponse >= 'a') AND (Reponse < Max_Min) THEN
               BEGIN                                      (* Minuscule     *)
               WRITE  (Reponse);
               Propose [i] := UPCASE (Reponse);
               i := i+1
               END;

               END; {Case}                             (* Fin de la saisie     *)
            END; {While}                               (* Propose []           *)
                                                       (*----------------------*)
         IF NOT Fin_Jeu THEN

(* Examen de la proposition *)
(*--------------------------*)

            BEGIN
            Fin_Jeu := Propose = A_Trouver;              (* Egalite avec la   *)
            IF Fin_Jeu THEN                              (* chaine mystere ?  *)
               WRITELN ('  Bravo, vous avez trouve')     (*-------------------*)
            ELSE

(* Analyse symbole par symbole *)
(*-----------------------------*)
```



```
            BEGIN
            Bien_Mis := 0;                          (* Bons symboles au  *)
            FOR i:= 1 TO Longueur DO                {  bon endroit        }
               IF Propose [i] = A_Trouver [i] THEN
                  Bien_Mis    := Bien_Mis + 1;
                                                    (* Bons symboles au  *)
            Bon_Symb := 0;                          {  mauvais endroit.   }
            FOR i := 1 TO Longueur DO               (* Boucle symboles   *)
               BEGIN                                {  mysterieux         }
               j := 1;
               REPEAT                               (* Boucle proposit.  *)
                  f := Propose [j] = A_Trouver [i];
                  IF f THEN                           { Si meme symbole   }
                     BEGIN
                     Bon_Symb := Bon_Symb + 1;        { le compter        }
                     Propose [j] := '*'               { l'eliminer de     }
                     END;                             { recherche future  }
                  j := j + 1;
                  UNTIL (j > Longueur) OR f;          { sortie de boucle  }
               END;

            WRITELN (Bon_Symb:3, ':', Bien_Mis);    (* Sortie resultats  *)
            Nb_Solut := Nb_Solut + 1
            END;

         END;
      UNTIL Fin_Jeu;                              (* Solution trouvee ou *)
   END;                                           (* abandon             *)

  WRITELN;
  WRITE   ('Voulez-vous arreter ? ');        (* Fin de boucle jeux ?      *)
  READ    (KBD, Propose [1]);                (*--------------------------*)
  UNTIL   (Propose[1] IN ['O','o']);

END.
```

Disque virtuel grâce à Debug

Notre série d'articles sur les fonctions du DOS nous a permis d'en apprendre beaucoups sur le DOS. Quelques données supplémentaires vont nous permettre de rajouter de nouvelles fonctions à l'initialisation, pour pouvoir gérer de nouveaux périphériques.

Jean-Paul PRUNIAUX

Dans votre manuel du DOS, vous avez un chapitre intitulé configuration ; celle-ci est l'« *ensemble des caractéristiques techniques d'un système informatique et de ses périphériques* » (Lexique de la micro-informatique par Pierre Le Beux — Sybex).

Et les périphériques (Devices en anglais) sont les divers appareils connectés à un ordinateur tels que écran, clavier, imprimantes, disques, même si ceux-ci sont en fait intégrés dans la carcasse de votre ordinateur.

En fait, tous ces périphériques sont surveillés, contrôlés et pilotés par des programmes spécifiques faisant partie du système.

Toutes ces caractéristiques peuvent être modifiées par le contenu du fichier CONFIG.SYS lu à l'initialisation de votre IBM PC.

Entre autres, la commande DEVICE = (nom de fichier) demande au DOS de charger et conserver en mémoire ce fichier qui va gérer un ou plusieurs périphériques.

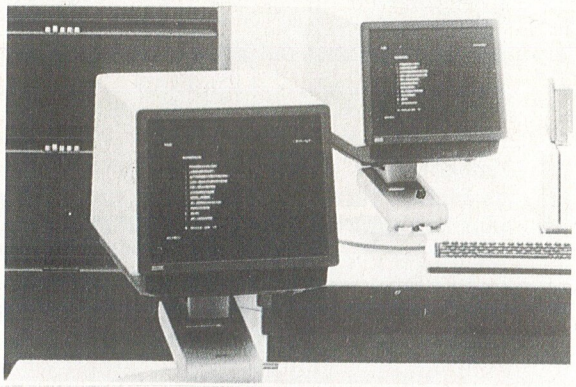
Dans nos numéros précédents, nous avons parlé à plusieurs des fonctions étendues de contrôle de clavier et d'écran du fichier ANSI.SYS, installé à l'initialisation dans le fichier CONFIG.SYS grâce à la commande DEVICE = ANSI.SYS.

tel programme va communiquer d'une manière bien précise avec le DOS, ce que nous allons voir dans cet article, au travers de l'écriture d'un disque virtuel résident en mémoire, mais d'abord regardons ce qu' est un disque virtuel.
Nous consulterons aussi le contenu du fichier ANSI.SYS à titre d'exemple.

Un disque virtuel, est une zone mémoire, avec des programmes de gestion de celle-ci faisant croire au système qu'il s'agit d'un disque. Sous réserve d'avoir de la mémoire disponible, cela permet d'avoir une unité disque supplémentaire bien pratique quand on a un PC avec un seul lecteur de disquette, avec des temps de « *lecture* » ou d'« *écriture* » divisé par 10, car le système n'aura pas à mettre en action la disquette ou le disque à chaque accès, bien pratique quand vous utilisez un programme dont une partie reste sur le disque et qui n'est chargé qu'en cas de besoin.

FORMAT D'UN FICHIER DEVICE

Dans le cadre du DOS 2.0 ce « *device* » doit pouvoir être appelé par le DOS pour éxécuter une fonction



parmi une liste de 12 fonctions (numérotées de 0 à 11, ou 0BH) et dans le cadre du DOS 3.0, 16 fonctions (numérotées de 0 à 16 ou 10H).

Cet appel se fait en 2 temps, tout d'abord, le DOS appelle le programme de « *stratégie* » avec dans les registres ES:BX l'adresse d'un bloc de paramètres contenant entre autre le code de la fonction que le « *device* » devra traiter ; puis dans un deuxième temps le DOS appèle le programme d'« *interruption* » qui doit traiter cette fonction.

l'appel de ces programmes se fait par le DOS avec un CALL FAR, c'est à dire en y mentionnant une adresse complète comprenant segment et déplacement. Ces sous-programmes doivent donc se terminer par un RETF (RETurn Far).

Ce découpage en 2 temps permettra d'inclure aisément la gestion du multitâche dans les versions futures du DOS (permettant ainsi à plusieurs programmes de se dérouler « *simultanément* » sous les yeux de l'utilisateur.)

EN TETE DU « DEVICE »

Ceci étant précisé, nous allons pouvoir examiner la première caractéristique que doit avoir ce fichier « *device* ». Il s'agit d'un entête de 12 octets permettant au DOS de savoir quel device, et combien, ce programme va permettre de gérer.

Tout d'abord, le fichier « device » peut contenir plusieurs programmes de gestion de périphériques. Si les 4 premiers octets contiennent FFH (FFFF:FFFF), alors il n'y a qu'un programme ou bien c'est le dernier de la liste. Si la valeur est différente, le premier mot (2 octets) contient l'adresse 16 bits par rapport au début de ce fichier d'un nouvel en tête, pour un second (troisième ou ...) programme de gestion de périphérique.

Examinons ceci sous DEBUG en vidant en mémoire l'entête du fichier ANSI.SYS :

```
A>DEBUG \ANSI.SYS
_D100 111
2D11:0100  FF FF FF FF 13 80 A2 00_AD 00 43 4F 4E 20 20 20   ......"._.CON
2D11:0110  20 20
_
```

FF FF FF FF, ANSI.SYS ne contient donc le programme de gestion que d'un seul périphérique.

Le mot suivant (2 octets) contient un code d'attribution de ce périphérique permettant de savoir ce que va faire ce programme. Les 16 bits de ce mot ont une signification comme suit :

Bit 15 = 1 si périphérique type caractère, 0 si du type bloc.
Bit 14 = 1 si IOCTL supporté.
Bit 13 (type bloc) = 1 si pas d'identificateur de table d'allocation de fichier (format différent du format IBM).
Bit 13 (type caractère et DOS 3.0) = 1 si sortie jusqu'au signal « occupé » (busy).
Bit 12 = réservé (doit être nul).
Bit 11 (DOS 3.0) = 1 si OPEN, CLOSE, RM supportés.
Bit 10 à 04 = réservé (doivent être nuls).
Bit 03 = 1 si gestion de l'horloge.
Bit 02 = 1 si périphérique NUL.
Bit 01 = 1 si gestion de la sortie standard (Ecran).
Bit 00 = 1 si gestion de l'entrée standard (Clavier).

Cette liste peut paraître rébarbative au nouveau venu, aussi examinons la ensemble :

Il existe 2 types de périphériques, ceux travaillant caractère par caractère (Clavier, écran, imprimante ...) et ceux travaillant bloc de données par bloc de données (les disques et disquettes) où le bloc de données est en fait un secteur. d'où ce premier bit 15 pour faire la différence entre ces 2 types de périphériques, la nature des informations passées n'étant pas la même.

IOCTL est un ensemble de fonctions du DOS que nous n'avons pas étudié, et que nous comptions plus spécialement aborder dans le cadre du DOS 3.0 (bien qu' une partie de celles-ci fassent déjà partie du DOS 2.0). Ce sigle signifie Input Output ConTroL, il s'agit donc des fonctions de contrôle des entrées-sorties. (Permet l'appel des fonctions 3 et 12 (0CH) du programme « *interruption* »).

Le bit 13 permet de savoir, dans le cadre du type bloc, si le disque est géré au format IBM ou non. La particularité principale du format IBM est l'identificateur de la table d'allocation de fichier. Nous reviendrons un peu plus loin sur cette table, il faut savoir qu'elle permet au DOS de connaître l'enchaînement physique des secteurs sur le disque constituant un même fichier. Quant à l'identificateur de cette table (son premier octet), nous en avons déjà parlé dans notre dernier article ; rappelons quand même ses valeurs :

- FFH = Double-Face, 8 secteurs par piste.
- FEH = Simple-Face, 8 secteurs par piste.
- FDH = Double-Face, 9 secteurs par piste.
- FCH = Simple-Face, 9 secteurs par piste.
- FBH = Double-Face,15 secteurs par piste.

Le bit 13 à 1, pour le DOS 3.0, et pour un périphérique du type caractère signifie que celui-ci va envoyer tous les caractères jusqu'à ce qu' il n'en ait plus ou que le périphérique lui réponde par le signal « Busy », c'est-à-dire qu'il est occupé avec les caractères déjà reçus et ne peut en recevoir d'autres. Sinon le « *device* » devra envoyer tous les caractères. Ce bit à 1 permet l'appel de la fonction numéro 16 (10H) du programme « *interruption* ».

Ceci permet de rester un minimum de temps dans ce sous-programme, rendant la main au programme appelant ou au système qui pourra faire autre chose en attendant que le périphérique en ait terminé.

Le bit 11, pour le DOS 3.0, permet l'accès aux fonctions de code 13, 14 et 15 (0DH, 0Eh et 0FH) du programme « *interruption* », permettant de déclarer des ouvertures (nouvelle série d'accès), fermetures et de vérifier s'il s'agit d'un périphérique fixe (disque dur) ou amovible (disquette).

Le bit 3 déclare un programme de gestion de l'horloge du DOS. Utile dans le cadre de l'adjonction d'une carte qui contiendrait par exemple une horloge permanente (qu'on ne serait pas obligé de remettre à l'heure à chaque re-mise sous tension).

Les bits 2,1 et 0 déclarent les programmes de gestion de NUL: et CON:.

Revenons au contenu de l'entête de ANSI.SYS (d'après un DOS 2.10), et examinons le mot suivant.

Nous lisons les octets 13H et 80H, formant ainsi le mot 8013H, ce qui décodé en binaire nous donne : 1000 0000 0001 0011.

Il s'agit d'un « *device* » caractère (bit 15 = 1), ne supportant pas les IOCTL (bit 14 = 0) ; ce n'est pas une horloge (bit 3 = 0), ni NUL: (bit 2 = 0), mais il s'agit du periphérique d'entrée standard (bit 1 = 1) et de sortie standard (bit 0 = 1).

Le mot suivant de l'entête contient l'adresse 16 bits du programme « *stratégie* » et le mot suivant celle du programme « *interruption* ». Notez qu' il s'agit d'adresse par rapport au début de l'entête du « *device* » !

Les adresses sont toutes considérées à partir de CS:0000, alors que pour un programme *.COM, le DOS rajoute au chargement en tête du programme un préfixe de segment de programme de 0100H octets ; de plus, lors du chargement d'un fichier, DEBUG le charge systématiquement en CS:0100.

Visualisons ces informations en désassemblant sous DEBUG le début des programmes « *stratégie* » et « *interruption* » de ANSI.SYS ; les octets suivants dans l'entête de ANSI.SYS sont A2 00 et AD 00, ce qui correspond aux mots 00A2H, et 00ADH, rajoutons les 0100H correspondant au PSP créé par DEBUG, ce qui nous donne 1A2 et 1AD :

```
_U 1A2
2D11:01A2 2E            CS:
2D11:01A3 891E9E00      MOV     [009E],BX
2D11:01A7 2E            CS:
2D11:01A8 8C06A000      MOV     [00A0],ES
2D11:01AC CB            RETF
2D11:01AD ...
_U 1AD
2D11:01AD 56            PUSH    SI
_
```

En CS:01A2, nous trouvons 2 instructions pour sauvegarder l'adresse du bloc de paramètres contenus dans BX et ES, puis le RETF. En CS:01AD, nous trouvons toute une suite d'instructions correspondant au programme « *interruption* » de ANSI.SYS.

Les 8 derniers octets de l'entête du « *device* » dans le cadre d'un « *device* » type caractère servent à déclarer son nom, par exemple CON, AUX, NUL ou bien tout autre nom que vous pourriez inventer (Pour le DOS ces noms sont aussi équivalent à CON:, AUX:, ...).

Dans le cadre d'un « *device* » type bloc, le premier de ces octets contient le nombre de périphériques gérés par ce « *device* ». La déclaration de 2 unités ferait gérer au DOS, en plus des unités A: et B: les 2 unités C: et D:.

Creéation de l'en tête :

Avec ce que nous venons d'apprendre, nous pouvons construire l'entête du programme de gestion de notre disque virtuel, à rajouter au système. Considérons un premier exemple de ce « *device* » que nous appellerons RAMDRV9.SYS, et que nous intégrerons au système par la commande de CONFIG.SYS : DEVICE=RAMDRV9.SYS. (RAM étant l'abréviation anglaise de mémoire vive, DRV celle de DRiVe, c'est-à-dire (programme de) pilotage et 9 car nous avons l'intention de créer un pseudo disque 9 secteurs par piste.

Pour commencer, initialisons DEBUG, en effet, nous désirons commencer à assembler à partir de l'adresse CS:0000. Modifions donc CS pour que cela soit possible, sans bouleverser le Préfixe de segment de Programme qu'a pu créer DEBUG, puis initialisons toute la zone mémoire dans laquelle nous créons notre code avec un F 1000:0000 L200 90, demandant de remplir la mémoire à partir de l'adresse CS:0000, sur une longueur de 200H octets avec le code 90H, qui correspond à l'instruction NOP (de ce fait, si nous n'assemblons pas une instruction en mémoire, le microprocesseur ne sera pas perdu).

Notre programme « *device* » ne contient qu 'un seul programme, les 4 premiers octets sont donc FF.

Le mot d'attribut est décomposé comme suit : bit 15 = 0, type bloc, bit 14 = 0, IOCTL non supporté, bit 13 = 1, format non IBM (pour des raisons de sécurité), tous les autres bits à 0. ce qui nous donne 0010 0000 0000 0000 soit 2000H.

Comme nous allons avoir besoin de place en mémoire, pour stocker d'autres informations et tables que nous verrons ultérieurement le programme « *stratégie* » ne commence qu'en CS:005A, immédiatement suivi par le programme « *interruption* ».

Comme mentionné plus haut, le programme « *stratégie* » récupère dans ES et BX l'adresse d'un bloc de paramètres que lui communique le DOS. Ce programme stocke en mémoire le contenu de ces registres, et sera ultérieurement exploité par le programme « *interruption* ». d'où la réservation des 2 mots en CS:0012 et les 2 instructions en CS:005A CS:MOV [Adresse],Registre stockant le contenu du registre mentionné en CS:adresse ; puis le RETF.

```
A>DEBUG
_RCS
CS 0D11
:1000
_F 1000:0000 L200 90
_A 0000
1000:0000  ;                    En Tete du device
1000:0000  DW     FFFF FFFF
1000:0004  DW     2000
1000:0006  DW     005A
1000:0008  DW     0065
1000:000A  DB     01 00 00 00 00 00 00 00
1000:0012
_A 0012
1000:0012 ;                     Adresse des parametres passes
1000:0012  DW     0000 0000
1000:0016
_
_A 005A
1000:005A ;                     Strategie
1000:005A  CS:MOV [0012],BX
1000:005F  CS:MOV [0014],ES
1000:0064  RETF
1000:0065
_A 0065
1000:0065 ;                     Interruption
1000:0065 (...)
_
```

APPELS DU « DEVICE »

Comme nous venons de voir dire, les fonctions du « *device* » sont appelées en 2 temps, d'abord l'adresse d'un bloc de paramètres est passée dans ES:BX au programme « *stratégie* », puis le traitement est fait par le programme « *interruption* » (tous deux devant se terminer par un RETF.

Ce bloc de paramètres comprend un en-tête constant, puis des paramètres additionnels en fonction de la demande faite par le DOS au « *device* ». Cet en tête constant est composé de 12 octets ayant les significations suivantes :

-[param] + 00 : Nombre d'octets constituant ce bloc de paramètres.
-[param] + 01 : Numéro de l'unité concernée par cette appel. Pour les « device » de type bloc, nous pouvons gérer plusieurs unités avec le même programme (voir l'octet 10 ou 0AH de l'en tête du « device »). Ce deuxième octet va nous préciser avec lequel nous travaillons.
-[param] + 02 : Code de la fonction demandée par le DOS.
-[param] + 03 : Mot d'état, à mettre à jour par le programme « interruption ».
-[param] + 05 : 8 octets resérvés par le DOS pour le futur.
-[param] + 0D : Paramètres supplémentaires dépendant de la fonction appelelée.

CODE DE LA FONCTION

Les fonctions que va appeler le DOS pour gérer le « *device* » sont les suivantes :

Pour les DOS 2.0 et 3.0 :

-00 : Initialisation.
-01 -type bloc- : Vérifie que la disquette a été changée.
-02 : Constitution d'un bloc de paramètres pour le BIOS.
-03 -si IOCTL- : Appel de la fonction

d'entrée d'IOCTL.
-04 : Lecture (Input).
-05 -type caractère- : Répond à la question « Est-ce qu'un caractère a été reçu ? » par Oui ou par Non, lit ce caractère s'il existe.
-06 -type caractère- : Etat de la mémoire-tampon de réception de caractères.
-07 -type caractère- : Réinitialise à zéro, la mémoire de réception de caractères.
-08 : Ecriture (Output).
-09 : Ecriture avec relecture de vérification.
-0A -type caractère- : Répond à la question « Y a-t-il un caractère prêt à être émis et pas encore parti ? » (Etat en écriture).
-0B -type caractère- : Destruction des caractères stockés en attente d'écriture.
-0C -si IOCTL- : Appel de la fonction de sortie d'IOCTL.

Pour le DOS 3.0 uniquement :

-0D -si bit 11 d'attribut du « *device* »- : Ouverture du périphérique.
-0E -si bit 11 d'attribut du « device»- : Fermeture du périphérique.
-0F -si bit 11 d'attribut du « device»—type bloc- : Répond à la question « S'agit-il d'un disque amovible ou d'un disque fixe ».
-10 -si bit 13 d'attribut du « *device* »—type car.- : Envoi des caractères possible jusqu'au signal 'occupé'.

Mot d' état :

Il s'agit du mot en [Paramètre] + 03 dans lequel le programme « *interruption* » met son compte-rendu avant de faire le RETF (le DOS l'a initialisé à 0000 avant l'appel). Il est en fait décomposé en 2 parties, les bits de poids fort donnant ce compte-rendu, et les bits de poids faible contenant éventuellement un code d'erreur.

Le bit 8 de ce mot doit signaler que l'opération demandée par le DOS a été faite.

Le bit 9 doit renvoyer une information « *busy* », c'est-à-dire que le périphérique est occupé (pour certaines fonctions, il sert parfois à transmettre une information légèrement différente). Il n'est modifié que par les fonctions concernant l'état, ou la fonction demandant si le disque est fixe ou amovible.

Les bits 10 à 14 sont pour le moment réservés et inutilisés.

Le bit 15 est à mettre à 1 en cas d' erreur, à ce moment là, les bits 0 à 7 (octet de poids faible de ce mot d'état) doivent être à charger avec un des codes d'erreur suivant :

-00 : Tentative de violation de la protection en écriture.
-01 : Unité inconnue.
-02 : Lecteur de disquette pas prêt.
-03 : Code de commande inconnu.
-04 : Erreur de contrôle de redondance cyclique (il s'agit d'un contrôle fait systèmatiquement grâce à des informations déposées sur le disque).
-05 : Structure demandée incompatible avec le périphérique.
-06 : Erreur de recherche.
-07 : Support inconnu.
-08 : Secteur pas trouvé.
-09 : Plus de papier sur l'imprimante.
-0A : Anomalie d'écriture.
-0B : Anomalie de lecture.
-0C : Problème général.
-0D : Reservé.
-0E : Réservé.
-0F : Changement de disque invalide.

DEBUT DU PROGRAMME « INTERRUPTION »

Pour nous aiguiller sur les programmes traitant les fonctions demandées par le DOS, nous allons nous servir d'une table contenant les adresses de tous ces programmes.

Parmi la liste des 12 fonctions ci-dessus, certaines ne peuvent théoriquement pas être appelées ; cependant si telle est le cas, elles sont traitées avec une sortie en erreur de code 3. Nous aborderons le traitement des autres fonctions plus loin. Tout ce que nous avons besoin de savoir pour le moment c'est qu' elles commencent aux adresses suivantes :

Fonction 00 : Init en CS:00A0.
Fonction 01 : Vérification du changement de disque en CS:0110.
Fonction 02 : Constitution d'un bloc de paramètres pour le BIOS en CS:120.
Fonction 04 : Lecture en CS:150.
Fonction 08 et 09 : Ecriture et ecriture avec vérification en CS:190. (La vérification a en fait été ignorée dans le cadre de notre exemple, en tenant compte du fait que la mémoire est sensée être fiable).

```
_A 0040
1000:0040 ;                    Table de traitement des fonctions
1000:0040 ;        Init Chek  BPB IOCT
1000:0040  DW      00A0 0110 0120 0079
1000:0048 ;
1000:0048 ;         Inp  ___  ___  ___
1000:0048  DW      0150 0079 0079 0079
1000:0050 ;
1000:0050 ;         Out VOut OSta OFls
1000:0050  DW      0190 0190 0079 0079
1000:0058 ;
1000:0058 ;        IOCT
1000:0058  DW      0079
1000:005A
```

Cette table étant créée, le début du programme « *interruption* » va l'exploiter pour s'aiguiller sur les instructions de traitement de ces fonctions.

Auparavant, le programme sauvegarde tous les registres qu'il est susceptible d'utiliser, nous devons en effet rendre les registres dans l'état où nous les avons trouvé. Cette sauvegarde est faite dans la pile par une série de PUSH.

Il faut savoir que la pile telle qu'elle est passée par le DOS peut contenir encore une vingtaine de PUSH (un CALL est équivalent à un PUSH et un CALL FAR à deux PUSH).

L'instruction CS:LDS BX,[0012] charge les registres DS et BX avec les 2 mots contenus à l'adresse CS:0012, le mot en CS:0012 étant chargé dans BX et le mot en CS:0014 dans DS. Ceci nous permet de récupérer dans DS:BX l'adresse du bloc de paramètres passés par le DOS au programme « *stratégie* ».

L'octet numéro 02 du bloc de paramètres est chargé dans AL par l'instruction suivante, il s'agit du code de fonction passé par le DOS. S'il est supérieur à 12 (0CH), nous terminons tout de suite en mettant le code d'erreur numéro 3 dans AL, remarquez que nous nous trouvons ici à l'adresse 0079, correspondant aux autres fonctions qui, entre 0 et 12 ne sont pas traitables par ce programme.

Si par contre le code de commande est inférieur à 0CH (JNA, Saut si Pas au dessus), nous nous retrouvons en CS:0093 pour nous aiguiller sur le bon traitement.

En CS:0080 nous trouvons le point de sortie en erreur commun à toutes les fonctions, le code d'erreur étant en AH, le mot d'état est complété dans AX en positionnant le code d'erreur.

Puis à partir de l'adresse CS:0082, nous trouvons la fin qui sera commune à toutes les fonctions. Celle-ci recharge DS et BX avec l'adresse du bloc de paramètres, y écrit le mot d'état contenu dans AX, puis dépile tous les registres avant de terminer par un RETF ; pour clore le traitement d'une fonction, il suffira de mettre dans AX le mot d'état et de faire un JMP 0082.

Notez que l'escargot utilisé dans les commentaires est un symbole voulant dire « *adresse* », généralement, « *adresse 20 bits, avec segment et déplacement* ».

```
_A 0065
1000:0065 ;                          Interruption
1000:0065  PUSH   DS
1000:0066  PUSH   ES
1000:0067  PUSH   AX
1000:0068  PUSH   BX
1000:0069  PUSH   CX
1000:006A  PUSH   DX
1000:006B  PUSH   SI
1000:006C  PUSH   DI
1000:006D  ;
1000:006D  CS:LDS BX,[0012]          ;DS:BX = @ Parametres
1000:0072  MOV    AL,[BX+02]         ;Code de commande
1000:0075  CMP    AL,0C              ;Inconnu ?
1000:0077  JNA    0093
1000:0079  MOV    AL,03                ;OUI, Code erreur
1000:007B
_A 0080
1000:0080 ;                          Fin, Erreur dans AL
1000:0080  MOV    AH,80
1000:0082
_A 0082
1000:0082 ;                          FIN ; AX contiend le compte rendu
1000:0082  CS:LDS BX,[0012]
1000:0087  MOV    [BX+03],AX
1000:008A  POP    DI
1000:008B  POP    SI
1000:008C  POP    DX
1000:008D  POP    CX
1000:008E  POP    BX
1000:008F  POP    AX
1000:0090  POP    ES
1000:0091  POP    DS
1000:0092  RETF
1000:0093
```

Arrivant en CS:0093, nous avons dans AH un code de commande compris entre 0 et 12 (0CH), il suffit de le transformer en mot dans AX, de le multiplier par 2, car les adresses de la tables d'aiguillage commencent tous les 2 octets (CS:0040, CS:0042, CS:0044,...), d'y ajouter 0040, adresse du début de la table. A la fin de ces opérations, nous avons dans SI l'adresse du programme de traitement de la commande obtenue par (2*Code) + 40H. Il nous suffit de faire un saut à l'adresse contenu dans le mot mémoire obtenu par le segment CS: et le contenu de SI. Ce que permet l'instruction CS:JMP [SI].

```
_A 0093
1000:0093  XOR     AH,AH          ;AX = Code de commande
1000:0095  SHL     AX,1           ;Fois 2 (Adresse)
1000:0097  MOV     SI,0040
1000:009A  ADD     SI,AX          ;Adresse de traitement
1000:009C  ;
1000:009C  CS:JMP [SI]
1000:009F
```

Etudions maintenant ce que doivent faire chacune des fonctions du « *device* ».

INIT

En plus de l'entête de 13 octets déjà décrit, le bloc de paramètres contient la place pour les informations suivantes qui sont à établir par le traitement de l'initialisation :

-[param.] + 0D : Nombre d'unités gérées par le « device ». (octet).
-[param.] + 0E : Adresse à partir de laquelle libérer la mémoire .(2 mots).
-[param.] + 12 : Adresse d'une table contenant les adresses 16 bits (mots) de bloc de paramètres pour le BIOS. (2 mots).
-[param.] + 16 : Pour le DOS 3.0, est un octet permettant d'identifier le periphérique. Il ne sert pas directement au DOS, mais pourra être recommuniqué par le DOS lors des futurs appels aux fonctions de ce « *device* ».

L'adresse à partir de laquelle libérer la mémoire va permettre de dire au DOS à partir d'où la mémoire : va ainsi rester en mémoire, le début de ce fichier jusqu'à l'adresse ainsi mentionnée.

BLOC DE PARAMETRES DU BIOS :

Les blocs de paramètres du BIOS sont des structures de données lui permettent de savoir comment gérer les périphériques de type bloc (inutiles pour le type caractère). Un bloc BPB est construit de la manière suivante :

-[BPB] + 00 :
Nombre d'octets par secteur (Mot).
-[BPB] + 02 :
Nombre de secteur par grappe (Octet).
-[BPB] + 03 :
Nombre de secteurs reservés (Mot).
-[BPB] + 05 :
Nombre de tables d'allocation de fichier (Octet).
-[BPB] + 06 :
Nombre maximum de fichiers par répertoire (Mot).
-[BPB] + 08 :
Nombre de secteurs (Mot).
-[BPB] + 0A :
Octet de description du périphérique (Octet).
-[BPB] + 0B :
Nombre de secteurs occupés par la table d'allocation de fichier (mot).

Toutes ces notions ont déjà été décrites dans notre article « Le DOS et ses disquettes » de la Revue de l'Utilisateur de l'IBM-PC n° 14. Résumons les ici :

-Piste : Surface vue sur le disque ou la disquette par la tête de lecture-écriture.
-Secteur : Portion de piste correspondant à un découpage angulaire de celle-ci
-Grappe (ou cluster) : Ensemble de secteurs (consécutifs) considérés comme une seule unité du point de vue du DOS.
-Table d'allocation de fichier (FAT) : table stockée sur le disque permettant, pour des fichiers s'étalant sur plusieurs secteurs, de connaître l'enchainement de ceux-ci.
-Octet de description du support : Octet codifié permettant de connaître la structure du support (disque, disquette, nombre de face ,...) ; dans le cadre d'un support au format IBM (bit 13 de l'attribut dans l'entête du « *device* »), cet octet est codifié de la même manière que l'identificateur de la table d'allocation de fichier, à savoir :

Bit 0 : 1 = double face, 0 = simple face.
Bit 1 : 1 = 8 secteurs par piste, 0 = différent de 8 secteurs par piste.
Bit 2 : 1 = Disque amovible, 0 = disque fixe.
Bit 3 à 7 : doivent être à 1.

Sinon, cet octet peut avoir la signification que désire le programmeur.
Notons aussi à propos du répertoire, que la place nécessaire dans celui-ci pour décrire un fichier s'étend sur 32 (20H) octets, pour pouvoir avoir 64 fichiers dans un répertoire (taille standard du DOS pour une disquette simple face, 9 secteurs par piste), le DOS va donc utiliser 32 x 64 = 2048 octets soit 4 secteurs (4 x 512).

Notre disque RAM étant là pour simuler une disquette simple face, 9 secteurs par piste, nous allons reproduire les caractéristiques standards du DOS pour une telle disquette, de la manière suivante :

-512 (200H) octets par secteurs.
-1 secteur par grappe.
-1 secteur réservé (le premier).
-2 FATs (qui seront stockées juste après le secteur réservé).
-64 (40H) comme capacité de fichier du répertoire.

-40 pistes x 9 secteurs soit 360 (168H).
-1111 1100 (FCH) comme description, c'est-à-dire disquette amovible, pas 8 secteurs par piste, pas double face.

Assemblons ceci en CS:0016, et juste derrière, construisons la table d'adresse des BPB ; en fait comme nous n'avons qu'une seule disquette à gérer, celle-ci ne contient qu'une adresse :

```
_A 0016
1000:0016  ;                    Bios Parameter Block
1000:0016  DW      0200
1000:0018  DB        01
1000:0019  DW      0001
1000:001B  DB        02
1000:001C  DW      0040
1000:001E  DW      0168
1000:0020  DB        FC
1000:0021  DW      0002
1000:0023
_A 0023
1000:0023 ;                     Adresse du BPB
1000:0023  DW      0016
1000:0025
```

FORMATTAGE DU DISQUE VIRTUEL

Le formatage, est l'opération qui consiste à préparer un support (disque, disquette, ...) pour lui permettre d'y lire ou d'y écrire des informations suivant un découpage défini. Ici, nous n'auront pas de formatage à proprement parler, mais il faut quand même préparer le disque virtuel pour que le DOS y reconnaisse des données qu'il s'attend à trouver dans les secteurs reservés, la FAT et le répertoire.

Le secteur reservé est toujours le secteur 0 de la piste 0 (le premier secteur du disque). Ce secteur commence par un programme lu au démarrage du système par le BIOS quand ce disque est dans l'unité A :. C'est ce programme qui va, entre-autre permettre au BIOS de charger le DOS.

Ce programme doit commencer par une instruction sur 3 octets (un JMP adresse) et continuer plus loin sur ce secteur, car à partir du quatrième octet, nous avons les données suivantes :

-8 octets pour mettre le nom du 'fabriquant' de la disquette..
-35 octets pour copie du BPB.

Puis, nous trouvons 3 informations facultatives :
-1 mot, le nombre de secteurs par piste.
-1 mot, le nombre de têtes de lecture (faces).
-1 mot, le nombre de secteurs cachés.

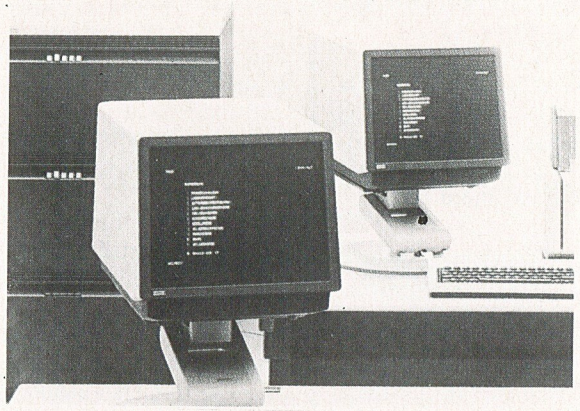
La table d'allocation de fichiers est une suite d'informations sur 12 bits (un octet et demi) ou 16 bits (2 octets) pour le disque dur. Appelons I la place de cette information dans la table, le contenu de cette information est le numéro de la grappe suivante faisant partie du même fichier.

Cependant un certain nombre de valeurs sont réservées, 000 indique une grappe non utilisée et FF8 à FFF indiquent des fins de chaînes de grappe.

Rajoutons à cela que le premier octet baptisé FAT ID (Identificateur de la FAT), est codifié avec la structure du périphérique, et que les 2 premières données sont réservées.

Il nous faudra donc initialiser la FAT de la manière suivante : (FAT ID), FF, FF, 00, 00, ...

Le répertoire va être complètement initialiser à 00 pour signaler qu' il n'y a aucun fichier sur ce disque.



PROGRAMME D'INITIALISATION

L'ensemble des instructions de notre « *device* » fait moins de 200H octets, aussi, nous allons faire démarrer le disque virtuel en CS:0200H, ce qui est équivalent à [CS + 0020]:0000. Ne connaissant pas la valeur de CS à priori, réservons nous en mémoire un mot pour stocker le segment où va commencer en mémoire le disque virtuel. Cette information nous permettra de calculer où se trouvent en mémoire les données correspondant à un secteur de numéro donné.

Puis nous allons définir en mémoire, le contenu du début du secteur que nous aurons à recopier en début de disque.

```
_A 0025
1000:0025  ;                    Segment du debut du disque v.
1000:0025  DW      0000
1000:0027
_A 0027
1000:0027  ;                    Pseudo enregistrement d' init
1000:0027  DB      00 00 00
1000:002A  DB      'JPP 2_86'
1000:0032  DW      0200
1000:0034  DB        01
1000:0035  DW      0001
1000:0037  DB        02
1000:0038  DW      0040
1000:003A  DW      0168
1000:003C  DB        FC
1000:003D  DW      0002
1000:003F
_
```

Nous en avons ici fini avec les réservations et définitions en mémoire. Passons maintenant à l'écriture de la fonction « INIT », dont l'adresse dans la table d'aiguillage est CS:00A0.

La première chose que nous faisons, est de calculer, grâce à DX, le segment où commence le disque virtuel en mémoire, et de le stocker en CS:0025.

Puis nous mettons à jour le bloc de paramètres que nous a communiqué le DOS, son adresse se trouve toujours dans les registres DS et BX.

Nous gérons avec ce « *device* » une nouvelle unité disque.

Nous calculons l'adresse à partir de laquelle libérer la mémoire en ajoutant le nombre de paragraphes pris par le disque virtuel en mémoire au segment précédemment calculé. Nous avons besoin de 512 octets x 9 secteurs x 40 pistes soit 184320 octets (2D000H octets), ou encore 2D00H paragraphes (un paragraphe égal 16 —10H— octets).

L'adresse à partir de laquelle libérer la mémoire est donc [CS + 0020 + 2D00]:0000.

La mise à jour du bloc de paramètres se termine en y mettant l'adresse CS:0023, contenant l'adresse 16 bits du BPB..

```
_A 00A0
1000:00A0  ;                    INIT
1000:00A0  PUSH    CS
1000:00A1  POP     DX                      ;Segment code
1000:00A2  ADD     DX,20                   ;Adresse debut disque
1000:00A5  CS:MOV  [0025],DX               ;Sauvegarde segment debut disque
1000:00AA  ;
1000:00AA  ;                    Mise a jour parametres
1000:00AA  ;
1000:00AA  MOV     BYTE PTR [BX+0D],01     ;Une unite
1000:00AE  ;
1000:00AE  ADD     DX,2D00                 ;2D00 paragraphes pour le disque
1000:00B2  MOV     WORD PTR [BX+0E],0000
1000:00B7  MOV     WORD PTR [BX+10],DX     ;Fin de memoire a conserver
1000:00BA  ;
1000:00BA  MOV     WORD PTR [BX+12],0023 ;
1000:00BF  MOV     WORD PTR [BX+14],CS     ;Adresse ou se trouve @BPB
```

Le « *formattage* » ne pose pas de problème, il suffit de suivre ce qui a été dit précédemment. La seule chose à savoir, est à quelle adresse mémoire correspond un secteur de notre disque virtuel.

Un secteur fait 512 octets, c'est-à-dire 200H. Donc la correspondance entre les secteurs est la suivante :

```
Secteur 000 = Disque¦0000 a Disque¦01FF, Secteur reserve.
Secteur 001 = Disque¦0200 a Disque¦03FF, 1ere FAT,
Secteur 002 = Disque¦0400 a Disque¦05FF, 1ere FAT suite.
Secteur 003 = Disque¦0600 a Disque¦07FF, 2eme FAT,
Secteur 004 = Disque¦0800 a Disque¦09FF, 2eme FAT suite.
Secteur 005 = Disque¦0A00 a Disque¦0BFF, debut du repertoire ...
```

Cette fonction se termine simplement en mettant dans AX le code d'état à mettre dans le bloc de paramètres et à faire un saut en CS:0084 comme vu au paragraphe précédent.

Nous en terminons ici, avec probablement ce qui est la fonction la plus laborieuse.

```
1000:00C2  ;                          ;Formate disque
1000:00C2  ;
1000:00C2  PUSH   CS
1000:00C3  POP    DS                  ;DS=CS
1000:00C4  MOV    SI,0027             ;DS:SI = pseudo code init
1000:00C7  MOV    ES,[0025]
1000:00CB  XOR    DI,DI               ;ES:DI = debut disque
1000:00CD  MOV    CX,0018
1000:00D0  CLD
1000:00D1  REP    MOVSB               ;Recopie du debut du secteur 00
1000:00D3  ;
1000:00D3  ES:MOV BYTE PTR [0200],FC  ;FAT ID
1000:00D9  ES:MOV BYTE PTR [0201],FF  ;2 elements reserves
1000:00DF  ES:MOV BYTE PTR [0202],FF
1000:00E5  MOV    DI,0203             ;ES:DI, secteur 00 + 3 octets
1000:00E8  XOR    AL,AL               ;AL = 00
1000:00EA  MOV    CX,03FD             ;jusque fin 2e secteur
1000:00ED  REP    STOSB               ;FAT = 00 00 ...
1000:00EF  ;
1000:00EF  PUSH   ES
1000:00F0  POP    DS                  ;DS=ES
1000:00F1  MOV    SI,0200             ;DS:SI = @ secteur 01
1000:00F4  MOV    DI,0600             ;ES:DI = @ secteur 03
1000:00F7  MOV    CX,0400             ;2 secteurs
1000:00FA  REP    MOVSB               ;recopie de la FAT
1000:00FC  ;
1000:00FC  MOV    DI,0A00             ;ES:DI = @ secteur 05
1000:00FF  MOV    CX,0800             ;4 secteurs
1000:0102  XOR    AL,AL               ;AL=00
1000:0104  REP    STOSB               ;Mise a 00 de DIR
1000:0106  ;
1000:0106  MOV    AX,0100
1000:0109  JMP    0082                ;FIN ; AX contiend le compte rendu
1000:010C
```

VÉRIFICATION DE SUPPORT

A la suite du bloc de paramètres communiqué par le DOS, nous trouvons :
-[param.] + 0D : Octet de description du périphérique, lu dans le BPB qui a été communiqué au DOS précédemment.
-[param.] + 0E : un octet dans lequel cette fonction doit répondre de la manière suivante :

01, le support (disque) n'a pas été changé.
00, état inconnu.
FF, le support a été changé.

Ayant affaire avec un disque virtuel en mémoire, la réponse tombe sous le sens, il ne peut avoir été changé, aussi mettons 01 en [Param.] + 0E.

En accord avec notre table d'aiguillage assemblons à l'adresse CS:0110 cette fonction.

```
_A 0110
1000:0110  ;                    MEDIA CHECK
1000:0110  ;
1000:0110  MOV    BYTE PTR [BX+0E],01
1000:0114  MOV    AX,0100
1000:0117  JMP    0082
1000:011A
```

CONSTRUCTION DE BPB

En plus des 13 octets d'entête connus du bloc de paramètres, nous trouvons :

[param.] + 0D : Octet de description du support du précédent Bloc de Paramètres du BIOS.
[param.] + 0E : Adresse 20 bits (Segment:déplacement) d'une zone de transfert pour les accès disque.
[param.] + 12 : Adresse 20 bits du nouveau Bloc de Paramètres pour le BIOS.

Le BPB a la même structure que celle qui a été décrite dans le paragraphe sur l'initialisation.

La zone de transfert, de la taille d'un secteur, est utilisée de manière différente, si nous avons affaire avec un « *device* » au format IBM ou non (bit 13 d'attribut du « *device* ».

Dans le cadre d'un disque au format IBM, le DOS y a préalablement chargé le premier secteur de la table d'allocation de fichiers (et le premier octet est donc le FAT ID), ce programme ne doit pas y toucher.

Dans le cas contraire, il s'agit d'une zone dont on est sensé pouvoir faire ce que l'on veut ; cependant, Micro-Soft préconise par contre la procédure suivante :

-Lire le secteur 000 dans cette zone mémoire.
-Déterminer si les premiers octets correspondent à un JMP sur 16 bits, auquel cas nous allons trouver un BPB sur le disque à partir du 3e octet.
-Si ce n'est pas le cas, il s'agit probablement d'un disque de version 1.x, auquel cas, servons nous du FAT ID pour générer le BPB.

Nous avions une zone mémoire en CS:0016 dans laquelle nous avons décrit un BPB pour l'initialisation. En sécurité, nous allons y réécrire les informations du secteur 000 (à partir de l'octet 0016).

En accord avec notre table d'aiguillage à partir de CS:0120, mettons dans ES la valeur du segment CS, dans DS le segment où commence notre disque virtuel (mémorisé en CS:0025), et chargeons les registres SI, DI et CX avec les valeurs adéquates pour le REP MOVSB.

Plutôt que d'écrire ces informations dans la zone de transfert, (pour suivre la procédure recommandée par *Micro-Soft*), nous allons modifier l'adresse de cette zone pour lui faire désigner le BPB.

Pour cela, nous rechargeons dans DS et BX l'adresse des paramètres passés au programme « *stratégie* » et mettons à jour les mots [param.] + 0E à [param.] + 14.

```
_A 0120
1000:0120  ;                    CONSTRUIT BPB
1000:0120  PUSH   CS
1000:0121  POP    ES
1000:0122  MOV    DI,16                  ;ES:DI = @ BPB
1000:0125  CS:MOV DS,[0025]
1000:012A  MOV    SI,000B                ;DS:SI = @ (Secteur 0, caract)
1000:012D  MOV    CX,000E                ;14 octets
1000:0130  CLD
1000:0131  REP    MOVSB                  ;Recopie octets 11 a 24
1000:0133  ;
1000:0133  CS:LDS BX,[0012]              ;DS:BX = @ Parametres
1000:0138  MOV    WORD PTR [BX+12],0016 ;
1000:013D  MOV    WORD PTR [BX+14],CS   ;@ BPB
1000:0140  MOV    WORD PTR [BX+0E],0016 ;
1000:0145  MOV    WORD PTR [BX+10],CS   ;@ DTA
1000:0148  ;
1000:0148  MOV    AX,0100
1000:014B  JMP    0082
1000:014E
```

LITTLE TRAMP CHARACTER LICENCED BY BUBBLES, INC., S.A.

RÉALISATION DUPUY-SAATCHI ÉQUIPEMENT

“Mon IBM PC AT et moi : quelle équipe musclée !”

Avec le dernier-né de la famille IBM PC, vous allez former une équipe vraiment très, très musclée.

Il a été conçu pour vous permettre de traiter une montagne d'informations, trois fois plus vite que les premiers ordinateurs personnels IBM.

L'IBM PC AT est proposé avec un disque fixe de 20 millions d'octets, ce qui vous donne assez de place pour saisir 10.000 pages d'informations ou retenir la liste de vos 200.000 meilleurs clients. Si vos besoins de stockage s'accroissent, il vous suffit d'ajouter un autre disque fixe de 20 millions d'octets.

Imprimante IBM 4201

Une telle richesse d'information demande a être protégée. C'est pourquoi l'IBM PC AT possède un verrouillage à clé.

Avec une mémoire vive de 256 kilo-octets ou plus, votre IBM PC AT vous permet d'accéder à des applications complexes, dans le domaine graphique par exemple. Et si les logiciels dont vous vous servez le permettent, vous pouvez pousser votre IBM PC AT jusqu'à 3 millions d'octets.

Conçu pour la performance, l'IBM PC AT ne rechigne pas pour autant aux travaux courants. Il accepte pratiquement tous les programmes écrits pour la famille IBM PC.

Appelez le (1) 47.22.22.22 pour avoir la liste des points de vente où vous pourrez essayer l'ordinateur personnel IBM AT et vous faire conseiller par votre distributeur agréé IBM PC.

IBM

LECTURE

Le plus dur a été fait. Maintenant, nous allons nous en sortir avec un peu de calcul, la taille des secteurs et la connaissance du bloc de paramètres communiqués par le DOS.

Le bloc de paramètres, que ce soit pour une écriture ou une lecture (fonctions 3,4,8,9,12 et 16) à la forme suivante :

[param.] + 0D : Octet de description du suuport du BPB.
[param.] + 0E : Adresse 20 bits de la zone de transfert des données (DTA).
[param.] + 12 : Nombre de secteurs (ou caractères suivant le type) à transférer. (mot).
[param.] + 14 : Secteur à partir duquel doit se faire le transfert (inutile pour le type caractère).

Dans tous les cas le « *device* » se doit de mettre à jour le nombre de secteurs (ou d'octets) qu'il a effectivement transféré.

Dans certains cas, le nombre de secteurs à transférer peut sembler aberrant (128 ou 80H) car il correspondrait à plus de 65 535 ou FFFFH octets (128 x 512 = 65 536, ou 80H x 200H = 10000H). En effet, on ne peut adresser plus d'octets sur le segment du DTA (DTA:0000 à DTA:FFFF). Cela est volontaire de la part de ceux qui ont écrit le DOS, et correspond à une optimisation de cette écriture. Dans un tel cas, la règle à suivre est d'ignorer les caractères au delà de 65 535 ou FFFFH.

Etudions maintenant les opérations à faire pour passer d'un numéro de secteur et d'un nombre de secteur à transférer à une adresse et à un nombre d'octets :

Nous avons mémorisé en CS:0025 le segment Disque: où démarre le disque virtuel ; et nous avons la correspondance suivante :

```
Secteur    Adresse
000        Disque¦0000
001        Disque¦0200
002        Disque¦0400
07F        Disque¦FE00
080        Disque¦10000, ce qui est impossible!
```

Il nous faut donc raisonner non pas à l'intérieur du même segment Disque:, mais calculer le segment où commence le secteur (Secteur) :

```
Secteur    Disque¦Adr.     Secteur¦0000
000        Disque¦0000     (Disque+0000)¦0000
001        Disque¦0200     (Disque+0020)¦0000
002        Disque¦0400     (Disque+0040)¦0000
043        Disque¦8600     (Disque+0860)¦0000
07F        Disque¦FE00     (Disque+0FE0)¦0000
080        Disque¦????     (Disque+1000)¦0000
081        Disque¦????     (Disque+1020)¦0000
```

La procédure de calcul de Secteur: est donc la suivante :
Secteur: = Disque: + (Numéro de secteur) * 0020H

(Nous vous rappelons qu'en base 16, 2*8 = 10H).

0020H s'exprime en binaire de la manière suivante : 10 0000. Donc multiplier par 0020H revient à décaler tous les bits du numéro de secteur 5 fois à gauche ; prenons un exemple (0020H = 32) :

Decimal	Hexadecimal	Binaire
67	043	0000 0100 0011
2144	860	1000 0110 0000

Pour calculer le nombre d'octets correspondant au nombre de secteurs, nous savons que nous avons 512 ou 200H octets par secteur. Il suffit donc de multiplier le nombre de secteurs par 512, soit 200H.

Pour cela, après avoir chargé le nombre de secteurs dans un registre mot, nous allons faire passer son poids faible en poids fort et mettre à zéro son poids faible, ceci reviendra à l'avoir multiplié par 100H. Puis nous ferons un décalage à gauche d'un bit pour le multiplier par 2. Voyons un exemple :

	Decimal	Hexa.	Binaire
	3	0003	0000 0000 0000 0011
Poids fort	771	0303	0000 0011 0000 0011
Poids faible	768	0300	0000 0011 0000 0000
Decalage	1536	0600	0000 0110 0000 0000

Si par hasard, nous avions un résultat supérieur à 10000H, nous retrouverions après le décalage le bit superflux dans l'indicateur de Carry. Il nous suffit donc de le tester pour éventuellement rectifier le compte au maximum possible : FFFFH.

Une fois ce calcul effectué, nous devons vérifier si nous ne dépassons pas la possibilité de stockage de caractères de la zone de transfert de donnée (DTA). Celles-ci étant stockées dans les registres ES et DI, il nous faut vérifier que DI + (Nombre d'octet) ne soit pas supérieur à FFFFH. Une addition sur ces 2 valeurs nous positionnera l'indicateur de Carry, si tel est le cas.

Nous devons avoir :
DI + (Nb Oct) < = FFFFH
ou encore :
(Nb Oct) < = FFFFH - DI

Donc si le Carry est positionné sur l'opération précédente, rectifions le nombre d'octets avec la valeur FFFFH - DI.

Cette analyse faite, nous pouvons assembler, toujours en accord avec notre table d'aiguillage, la fonction de lecture en CS:0150 :

L'opération consiste ici, grâce à l'analyse ci-dessus, à préparer les registres ES, DI, DS, SI et CX pour l'instruction REP MOVSB qui assurera le transfert de donnée.

Rappelons à propos de cette instruction que DS:SI contient l'adresse où doivent être lus les octets, ES:DI contient l'adresse où écrire les octets et CX contient le nombre d'octets à transférer.

```
_A 0150
1000:0150  ;                    READ
1000:0150  ;
1000:0150  LES    DI,[BX+0E]          ;ES:DI = @ ou ecrire
1000:0153  MOV    DX,[BX+14]          ;1er secteur a lire (<360 ou 168H)
1000:0156  MOV    CL,05
1000:0158  SHL    DX,CL               ;Mult par 20H (paragraphes)
1000:015A  CS:ADD DX,[0025]           ;DX: = segment ou lire
1000:015F  ;
1000:015F  MOV    CX,[BX+12]          ;CX = Nombre de secteur
1000:0162  MOV    CH,CL               ;CH =   "         "
1000:0164  XOR    CL,CL               ;CX = nb sec * 100H
1000:0166  SHL    CX,1                ;transforme en octets
1000:0168  JNC    016D                ;Plus de 10000H octets ?
1000:016A  MOV    CX,FFFF               ;Oui ! CX = maxi
1000:016D
_A 016D
1000:016D  MOV    AX,CX               ;Verifie depassement DTA
1000:016F  ADD    AX,DI
1000:0171  JNC    0178                ;Au dela de ES:DI ?
1000:0173  MOV    CX,FFFF               ;OUI ! CX = maxi
1000:0176  SUB    CX,DI                 ;         _ depart
1000:0178
_A 0178
1000:0178  MOV    DS,DX
1000:017A  XOR    SI,SI               ;DS:SI = @ debut secteur
1000:017C  CLD
1000:017D  REP    MOVSB
1000:017F  ;
1000:017F  MOV    AX,0100
1000:0182  JMP    0082
1000:0185
```

ÉCRITURE

Vu ce qui a été dit, à propos de la lecture, nous n'avons pas grand chose à ajouter, si ce n'est que nous avons 2 fonctions d'écriture.

La première (code 08) fonctionne de la même manière que la fonction de lecture précédemment décrite, la seconde (code 09) quant à, elle se doit après l'écriture de refaire une lecture pour vérifier l'éxactitude des données écrites.

Cette deuxième fonction est appelée par le DOS à la place de la fonction 08 si la commande VERIFY est ON.

En fait dans notre exemple, nous avons ignoré ceci, en faisant confiance à la mémoire.

La programme est ici très similaire à ce qui a déjà été écrit, la différence étant que les informations sources (DS:SI) à transéférer se trouvent dans le DTA et l' endroit où les écrires est une adresse correspondant à un secteur du disque virtuel (ES:DI).

```
_A 0190
1000:0190  ;                    WRITE
1000:0190  ;
1000:0190  MOV     DX,[BX+14]            ;1er secteur
1000:0193  MOV     CL,05
1000:0195  SHL     DX,CL                 ;Mult par 200H
1000:0197  CS:ADD  DX,[0025]
1000:019C  ;
1000:019C  MOV     CX,[BX+12]            ;Nb  secteur
1000:019F  MOV     CH,CL
1000:01A1  XOR     CL,CL
1000:01A3  SHL     CX,1
1000:01A5  JNC     01AA                  ;Plus de 10000H octets ?
1000:01A7  MOV     CX,FFFF                 ;Oui ! CX = maxi
1000:01AA
_A 01AA
1000:01AA  LDS     SI,[BX+0E]            ;DS:SI = @ Ou  lire
1000:01AD  ;
1000:01AD  MOV     AX,CX                 ;Verifie depassement DTA
1000:01AF  ADD     AX,SI
1000:01B1  JNC     01B8                  ;Au dela de DS:SI ?
1000:01B3  MOV     CX,FFFF                 ;OUI ! CX = maxi
1000:01B6  SUB     CX,SI                   ;         _ depart
1000:01B8
_A 01B8
1000:01B8  MOV     ES,DX
1000:01BA  XOR     DI,DI                 ;ES:DI = @ debut secteur
1000:01BC  CLD
1000:01BD  REP     MOVSB
1000:01BF  ;
1000:01BF  MOV     AX,0100
1000:01C2  JMP     0082
1000:01C5
```

Il ne nous reste plus qu'à sauvegarder notre programme sous le nom RAMDRV9.SYS par exemple :

```
_N RAMDRV9.SYS
_RCX
CX 0000
:1C5
_W 1000:0000
Ecriture 01C5 octets
_Q
A>
```

Pour utiliser ce disque virtuel, il nous suffit de rajouter la ligne suivante dans le fichier CONFIG.SYS

```
DEVICE=RAMDRV9.SYS
```

Si ce fichier n'existe pas, créons le ensemble :

```
A>COPY CON CONFIG.SYS
DEVICE=RAMDRV9.SYS
(Ctrl Break)
A>
```

Maintenant, réinitialisons notre PC en appuyant simultanément sur les touches Ctrl, Alt et Del.

Après cette réinitialisation, si votre PC ne comporte qu'une ou deux disquettes, vous allez en fait avoir accès à 3 disquettes. En effet, que vous ayez une ou deux disquettes, le DOS gère 2 unités A: et B: dans le même support physique, vous demandant de changer de disquette si nécessaire ; maintenant, vous avez une troisième disquette C:, et commencez à vous familiariser avec, en tapant quelques commandes telles que :

```
A>DIR C:

 Volume dans unite C n'a pas de label
 Repertoire de  C:\

Fichier introuvable

A>COPY WS*.* C/
A>C:
C>DIR

 Volume dans unite C n'a pas de label
 Repertoire de  C:\

WS        COM    25600  17/12/84  19:10
WSMSGS    OVR    33792   1/01/80   1:51
WSOVLY1   OVR    46080   1/01/80   0:59
        3 Fichier(s)
    22016 octets disponibles

C>
```

UN AUTRE EXEMPLE

En fin de cet article, vous allez trouver le listing d'un autre exemple de disque virtuel, construit exactement de la même manière, mais ayant un capacité de stockage inférieure à celui qui nous a servi de support dans la rédaction de cet article.

En effet, notre disque virtuel de 9 secteurs par piste et 40 secteurs, encombre quand même 40 * 9 * 512 octets, soit 184 320 octets (180 ko). Nous avons donc modifié quelques paramètres, (et ne lui avons mis qu'une FAT) de façon à ce que vous ayez un deuxième exemple de 256 secteurs, ce qui nous fait une capacité de 256 * 512 octets soit 131 072 octets (130 ko).

LES AUTRES FONCTIONS

Nous venons de passer en revue toutes les fonctions que nous avions besoin de traiter, dans le cadre de notre disque virtuel, complétons cependant en décrivant les autres fonctions existantes, et leur mode de communication avec le DOS.

ECRITURES, LECTURES

Toutes les fonctions d'écriture et de lecture, correspondant aux codes 3,4,8,9,12 et 16 reçoivent le même bloc de paramètres du DOS, été décrit dans le chapitre LECTURE.

La différence entre les « *devices* » de type bloc et ceux de type caractères résident dans la signification du mot [param.] + 12. Dans le premier cas, il s'agit d'un nombre de secteurs, dans le deuxième cas, il s'agit d'un nombre d'octets ou caractères.

FONCTION 5

Cette fonction répond à la question « Un caractère a-t-il été reçu ? ».

En plus des 13 octets d'entête standard, nous trouvons un 14e octet : [param.] + 0D : Octet éventuelllement lu.

Si aucun octet n'est arrivé, le programme répond « *Done* » + « *Busy* » dans le mot d'état de l'entête (bit 8 et 9 à 1).
Si un octet est arrivé, le programme le conserve pour les prochaines lectures par la fonction 4 (ou autre), mais en délivre une copie en [param.] + 0D ; puis il répond « *done* » et « *not busy* » dans le mot d'état de l'entête (bit 8 à 1, et bit 9 à 0).

FONCTION 6

Cette fonction ne comprend pas de paramètre supplémentaire, et répond dans le bit « *busy* » du mot d'état à la question : « *Y a-t-il un caractère déjà reçu et disponible venant de ce périphérique* ».

Si tel est le cas, le programme doit

répondre « *done* » et « *not busy* » ; signalant au DOS que la prochaine lecture se fera d'près la mémoire de réception de ce programme.

Sinon, le programme répondra par « *done* » et « *busy* », la prochaine lecture devant se faire en attendant un caractère.

FONCTION 10 OU 0AH

Cette fonction ne comprend pas de paramètre supplémentaire non plus et répond dans le bit « *busy* » à la question : « Y a-t-il une opération d'écriture en cours ? ».

Si tel est le cas, ce programme répond « done » et « *busy* » dans le mot d'état des paramètres ; signalant au DOS que la prochaine demande d'écriture devra attendre la fin de celle-ci.

Sinon, le programme devra répondre « *done* » et « *not busy* », signalant au DOS que la prochaine demande d'écriture partira directement vers le périphérique.

DESTRUCTION DE MEMOIRE TAMPON

Que ce soit en émission, ou en réception, la plupart des gestionnaires de périphériques se servent des possibilités des interruptions en utilisant une mémoire-tampon servant à recueillir les caractères en réception, en attendant que le programme principal les puise un par un, et avec une mémoire-tampon dans laquelle tout le texte d'un message à envoyer va être stocké alors que les caractères ne vont partir qu'un par un.

Les fonctions 7 (pour la lecture) et 11 ou 0BH (pour l'écriture) permettent de détruire ces mémoires-tampon.

Aucun paramètre supplémentaire n'est passé, et le programme répond « *done* » quand le contenu de ces deux mémoires-tampon a été détruit.

OUVERTURE ET FERMETURE

Ces fonctions 13 ou 0Dh et 14 ou 0EH sont appelées par le DOS 3.0 si le bit 11 du mot d'attribut dans l'entête du « *device* » est mis à 1.

Elles permettent au DOS d'informer le « *device* » de l'activité qu'il a à gérer, permettant de compter le nombre de fichiers en cours d'accès ; avec une destruction de toutes les mémoires-tampon par exemple lorsque ce compte retombe à 0.

Elles peuvent aussi permettre d'envoyer une chaîne d'initialisation en cas de sortie sur imprimante pour télécharger par exemple la hauteur de page, la police de caractères, ...

Remarquez que les périphériques CON:, AUX: et PRN: sont toujours ouverts par le DOS.

Le bloc de paramètres passé par le DOS ne comporte aucun paramètre supplémentaire.

SUPPORT AMOVIBLE

Cette fonction numéro 15 appelée par le DOS 3.0 si le bit 11 du mot d'attribut du « *device* » est mis à 1 répond à la question : « Le disque (disquette) est-il amovible ? ».

Ce programme ne comporte aucun paramètre supplémentaire dans l'entête, est répond à la question par « *done* » et « *busy* » si le disque n'est pas amovible (fixe).

Il va par contre répondre par « *done* » et « *not busy* » si ce disque est amovible.

Le DOS assume que cette fonction va toujours réussir, et ne testera pas le bit d'erreur du mot d'état dans le bloc de paramètre au retour de cette fonction.

```
A>DEBUG
_F 0000 L200 90
_RCS
CS 2D11
:3000
_A 0000
3000:0000  ;                    En Tete du device
3000:0000  DW     FFFF FFFF
3000:0004  DW     2000
3000:0006  DW     005A
3000:0008  DW     0065
3000:000A  DB     01 00 00 00 00 00 00 00
3000:0012
_A 0012
3000:0012 ;                     Adresse des parametres passs
3000:0012  DW     0000 0000
3000:0016
_A 0016
3000:0016  ;                    Bios Parameter Block
3000:0016  DW     0200
3000:0018  DB       01
3000:0019  DW     0001
3000:001B  DB       01
3000:001C  DW     0040
3000:001E  DW     0100
3000:0020  DB       F8
3000:0021  DW     0002
3000:0023
_A 0023
3000:0023 ;                     Adresse du BPB
3000:0023  DW     0016
3000:0025
_A 0025
3000:0025  ;                    Segment du debut du disque v.
3000:0025  DW     0000
3000:0027
_A 0027
3000:0027  ;                    Pseudo enregistrement d´ init
3000:0027  DB     00 00 00
3000:002A  DB     ´JPP 2_86´
3000:0032  DW     0200
3000:0034  DB       01
3000:0035  DW     0001
3000:0037  DB       01
3000:0038  DW     0040
3000:003A  DW     0100
3000:003C  DB       F8
3000:003D  DW     0002
3000:003F
_A 0040
3000:0040 ;                     Table de traitement des fonctions
3000:0040 ;       Init Chek  BPB IOCT
3000:0040  DW     00A0 0100 0110 0080
3000:0048 ;
3000:0048 ;        Inp  ___  ___  ___
3000:0048  DW     0140 0080 0080 0080
3000:0050 ;
3000:0050 ;        Out VOut OSta OFls
3000:0050  DW     0180 0180 0080 0080
3000:0058 ;
3000:0058 ;       IOCT
3000:0058  DW     0080
3000:005A
_A 005A
3000:005A ;                     Strategie
3000:005A  CS:MOV [0012],BX
3000:005F  CS:MOV [0014],ES
3000:0064  RETF
3000:0065
_A 0065
3000:0065 ;                     Interruption
3000:0065  PUSH   DS
3000:0066  PUSH   ES
3000:0067  PUSH   AX
3000:0068  PUSH   BX
3000:0069  PUSH   CX
3000:006A  PUSH   DX
3000:006B  PUSH   SI
3000:006C  PUSH   DI
3000:006D  ;
3000:006D  CS:LDS BX,[0012]     ;DS:BX = @ Parametres
3000:0072  MOV    AL,[BX+02]    ;Code de commande
3000:0075  CMP    AL,0C         ;Inconnu ?
3000:0077  JNA    0093
3000:0079  MOV    AL,03           ;OUI, Code erreur
3000:007B
_A 0080
3000:0080 ;                     Fin, Erreur dans AL
3000:0080  MOV    AH,80
3000:0082
_A 0082
3000:0082 ;                     FIN ; AX contiend le compte rendu
3000:0082  CS:LDS BX,[0012]
3000:0087  MOV    [BX+03],AX
3000:008A  POP    DI
3000:008B  POP    SI
3000:008C  POP    DX
3000:008D  POP    CX
3000:008E  POP    BX
3000:008F  POP    AX
3000:0090  POP    ES
3000:0091  POP    DS
3000:0092  RETF
3000:0093
_A 0093
3000:0093  XOR    AH,AH         ;AX = Code de commande
3000:0095  SHL    AX,1          ;Fois 2 (Adresse)
3000:0097  MOV    SI,0040
3000:009A  ADD    SI,AX         ;Adresse de traitement
3000:009C  ;
```

```
3000:009C  CS:JMP [SI]
3000:009F
_A 00A0
3000:00A0  ;                    INIT
3000:00A0  PUSH   CS
3000:00A1  POP    DX                    ;Segment code
3000:00A2  ADD    DX,1C                 ;Adresse debut disque
3000:00A5  CS:MOV [0025],DX             ;Sauvegarde segment debut disque
3000:00AA  ;
3000:00AA  ;                    Mise a jour parametres
3000:00AA  ;
3000:00AA  MOV    BYTE PTR [BX+0D],01   ;Une unite
3000:00AE  ;
3000:00AE  ADD    DX,2000               ;2000 paragraphes pour le disque
3000:00B2  MOV    WORD PTR [BX+0E],0000
3000:00B7  MOV    WORD PTR [BX+10],DX   ;Fin de memoire a conserver
3000:00BA  ;
3000:00BA  MOV    WORD PTR [BX+12],0023 ;
3000:00BF  MOV    WORD PTR [BX+14],CS   ;Adresse ou se trouve @BPB
3000:00C2  ;
3000:00C2  ;                    ;Formate disque
3000:00C2  ;
3000:00C2  PUSH   CS
3000:00C3  POP    DS                    ;DS=CS
3000:00C4  MOV    SI,0027               ;DS:SI = pseudo code init
3000:00C7  MOV    ES,[0025]
3000:00CB  XOR    DI,DI                 ;ES:DI = debut disque
3000:00CD  MOV    CX,0018
3000:00D0  CLD
3000:00D1  REP    MOVSB                 ;Recopie du dbut du secteur 00
3000:00D3  ;
3000:00D3  ES:MOV BYTE PTR [0200],FC    ;FAT ID
3000:00D9  ES:MOV BYTE PTR [0201],FF    ;2 elements reserves
3000:00DF  ES:MOV BYTE PTR [0202],FF
3000:00E5  MOV    DI,0203               ;ES:DI, secteur 01 + 3 octets
3000:00E8  XOR    AL,AL                 ;AL = 00
3000:00EA  MOV    CX,03FD               ;jusque fin secteur 02
3000:00ED  REP    STOSB                 ;FAT = 00 00 ...
3000:00EF  ;
3000:00EF  PUSH   ES
3000:00F0  POP    DS                    ;DS=ES
3000:00F1  ;
3000:00F1  MOV    DI,0400               ;ES:DI = @ secteur 03
3000:00F4  MOV    CX,0800               ;4 secteurs
3000:00F7  XOR    AL,AL                 ;AL=00
3000:00F9  REP    STOSB                 ;Mise a 00 de DIR
3000:00FB  ;
3000:00FB  MOV    AX,0100
3000:00FE  JMP    0082                  ;FIN ; AX contiend le compte rendu
3000:0100
```

```
_A 0100
3000:0100  ;                    MEDIA CHECK
3000:0100  ;
3000:0100  MOV    BYTE PTR [BX+0E],01
3000:0104  MOV    AX,0100
3000:0107  JMP    0082
3000:010A
_A 0110
3000:0110  ;                    CONSTRUIT BPB
3000:0110  PUSH   CS
3000:0111  POP    ES
3000:0112  MOV    DI,16                 ;ES:DI = @ BPB
3000:0115  CS:MOV DS,[0025]
3000:011A  MOV    SI,000B               ;DS:SI = @ (Secteur 0, caract)
3000:011D  MOV    CX,000E               ;11 octets
3000:0120  CLD
3000:0121  REP    MOVSB                 ;Recopie octets 11 a 24
3000:0123  ;
3000:0123  CS:LDS BX,[0012]             ;DS:BX = @ Parametres
3000:0128  MOV    WORD PTR [BX+12],0016 ;
3000:012D  MOV    WORD PTR [BX+14],CS   ;@ BPB
3000:0130  MOV    WORD PTR [BX+0E],0016 ;
3000:0135  MOV    WORD PTR [BX+10],CS   ;@ DTA
3000:0138  ;
3000:0138  MOV    AX,0100
3000:013B  JMP    0082
3000:013E
_A 0140
3000:0140  ;                    READ
3000:0140  ;
3000:0140  LES    DI,[BX+0E]            ;ES:DI = @ ou ecrire
3000:0143  MOV    DX,[BX+14]            ;1er secteur a lire (<360 ou 168H)
3000:0146  MOV    CL,05
3000:0148  SHL    DX,CL                 ;Mult par 20H (paragraphes)
3000:014A  CS:ADD DX,[0025]             ;DX: = segment ou lire
3000:014F  ;
3000:014F  MOV    CX,[BX+12]            ;CX = Nombre de secteur
3000:0152  MOV    CH,CL                 ;CH =    "          "
3000:0154  XOR    CL,CL                 ;CX = nb sec * 100H
3000:0156  SHL    CX,1                  ;transforme en octets
3000:0158  JNC    015D                  ;Plus de 10000H octets ?
3000:015A  MOV    CX,FFFF                 ;Oui ! CX = maxi
3000:015D
_A 015D
3000:015D  MOV    AX,CX                 ;Verifie depassement DTA
3000:015F  ADD    AX,DI
3000:0161  JNC    0168                  ;Au dela de ES:DI ?
3000:0163  MOV    CX,FFFF                         ;OUI ! CX = maxi
3000:0166  SUB    CX,DI                   ;         _ depart
3000:0168
```

```
_A 0168
3000:0168  MOV   DS,DX
3000:016A  XOR   SI,SI                ;DS:SI = @ debut secteur
3000:016C  CLD
3000:016D  REP   MOVSB
3000:016F  ;
3000:016F  MOV   AX,0100
3000:0172  JMP   0082
3000:0175
_A 0180
3000:0180  ;                     WRITE
3000:0180  ;
3000:0180  MOV   DX,[BX+14]           ;1er secteur
3000:0183  MOV   CL,05
3000:0185  SHL   DX,CL                ;Mult par 200H
3000:0187  CS:ADD DX,[0025]
3000:018C  ;
3000:018C  MOV   CX,[BX+12]           ;Nb  secteur
3000:018F  MOV   CH,CL
3000:0191  XOR   CL,CL
3000:0193  SHL   CX,1
3000:0195  JNC   019A                 ;Plus de 10000H octets ?
3000:0197  MOV   CX,FFFF                    ;Oui ! CX = maxi
3000:019A
_A 019A
3000:019A  LDS   SI,[BX+0E]           ;DS:SI = @ Ou  lire
3000:019D  ;
3000:019D  MOV   AX,CX                ;Verifie depassement DTA
3000:019F  ADD   AX,SI
3000:01A1  JNC   01A8                 ;Au dela de DS:SI ?
3000:01A3  MOV   CX,FFFF                ;OUI ! CX = maxi
3000:01A6  SUB   CX,SI                  ;       _ depart
3000:01A8
_A 01A8
3000:01A8  MOV   ES,DX
3000:01AA  XOR   DI,DI                ;ES:DI = @ debut secteur
3000:01AC  CLD
3000:01AD  REP   MOVSB
3000:01AF  ;
3000:01AF  MOV   AX,0100
3000:01B2  JMP   0082
3000:01B5
_N RAMDRVO.SYS
_RCX
CX 0000
:1C5
_W 1000:0000
Ecriture 01C5 octets
_Q
A>COPY CON CONFIG.SYS
DEVICE=RAMDRVO.SYS
(Ctrl Break)
A>
```


En matière de distribution l'union fait la marge

Vous distribuez les grands noms du « soft », Wordstar, Symphony, Textor, Open Access, Multiplan... et bien d'autres.

La distribution est un métier difficile.

Les intermédiaires dont trop nombreux.

Résultat : vos marges en font les frais.

Et ce n'est pas normal.

Vous vous doutez qu'il est possible de faire mieux.

Mais comment ?

Nous créons, pour vous une chaîne de distributeurs indépendants avec comme objectif de vous permettre de dégager des marges supérieures, par des remises plus importantes et d'augmenter vos ventes puisque cette chaîne de distributeurs indépendants vous assure une couverture publicitaire gratuite, ainsi qu'une assistance marketing permanente.

L'union fait la marge dans un réseau d'indépendants en micro-informatique

N'hésitez pas,
Ecrivez ou mieux téléphonez à
Philippe Gélin ou Jean-Michel Humeau
43.27.66.72

ATHEMA CONTACT
27, rue de l'Armorique
75015 Paris

NOTE A PROPOS DE L'ASSEMBLEUR

Comme précédemment, voici la signification des instructions assembleur employée dans cet article :

; commentaire : Le texte qui suit les 2 points est ignoré de DEGUG quand nous sommes en mode Assemblage (génère cependant une erreur sur les DB et DW.

ADD AX,DI : Additionne au contenu de AX le contenu de DI.

ADD DX,2D00 :Additionne au contenu de DX la valeur 2D00H.

CS:ADD DX,[0025] : Additionne au contenu de DX le contenu du mot mémoire à l'adresse CS:0025 (Voir CS:).

CLD : Mise à zéro de l'indicateur de direction (CLear Direction flag). Cette indicateur à zéro assure l'autodécrémentation de SI et/ou DI lors des instructions de manipulation de chaine (MOVSB par exemple).

CMP AL,0C : Compare le contenu de AL avec la valeur 0CH, en faisant la soustraction AL - 0C pour positionner les indicateurs arithmétiques et logiques.

CS: : La plus part des instructions adressant la mémoire utilisent le registre segment DS: par défaut. CS: précédent une instruction demande d'utiliser à la place le registre CS:.

DB 'JPP 2-86' : Definition d'octets (Bytes) en mémoire par la suite des codes ASCII du texte 'JPP 2-86'.

DB 01 00 00 : Definition d'octets en mémoire par la suite des codes héxadécimaux.

DW 00A0 0110 0120 0079 : Definition en mémoire de mots (Word) par la suite des codes héxadécimaux.

JMP 0082 : Saut à l'adresse CS:0082.

CS :JMP [SI] : Saut à l'adresse calculée avec à partir du contenu du mot en mémoire en CS:SI.

JNA 0093 : Saut en CS:0093 si pas au dessus (Not Above).

JNC 016D : Saut en CS:016D si pas de Carry (ou encore Saut si inférieur strictement).

LDS SI,[BX+0E] : Charge les registres DS et SI avec les informations en mémoire contenue à l'adresse calculée (sur DS:) au moyen du contenu de BX auquel on rajoute 0EH. Si BX contient 100H, SI sera chargé avec le mot contenu en DS:010EH et DS sera chargé avec le mot contenu à l'adresse DS:0110H.

CS :LDS BX,[0012] : Charge les registres DS et BX avec l'adresse contenu en CS:0012, BX est chargé avec le mot en CS:0012 et DS avec le mot en CS:0014.

LES DI,[BX+0E] : Charge les registres ES et DI avec les informations contenues en DS:[BX+0E], voir LDS SI,[].

MOV AX,0100 : Met dans AX (mot) la valeur 0100H.

MOV AH,80 : Met dans AH (octet) la valeur 80H.

MOV AX,CX : Met dans AX le contenu de CX (mot).

MOV CH,CL : Met dans CH le contenu de CL (octet).

MOV ES,[0025] : Met dans ES le mot contenu en mémoire à l'adresse DS:0025.

CS :MOV DS,[0025] : Met dans DS le mot contenu en mémoire à l'adresse CS:0025 (Voir CS:).

MOV [009E],BX : Met en mémoire à l'adresse DS:009E le mot contenu dans BX.

CS :MOV [0012],BX : Met en mémoire à l'adresse CS:0012 le mot contenu dans BX (Voir CS:).

MOV CX,[BX+12] : Met dans le registre CX le mot venant de la mémoire dont l'adresse (sur DS:) est calculée en ajoutant 12H au contenu de BX.

MOV AL,[BX+02] : Met dans le registre AL l'octet mémoire dont l'adresse (sur DS:) est calculée en ajoutant 02H au contenu de BX.

MOV [BX+03],AX : Met en mémoire, à l'adresse (Sur DS:) calculée en ajoutant 03H au contenu de BX, le mot contenu dans AX.

MOV WORD PTR [BX+10],CS : Met en mémoire à l'adresse (sur DS:) calculée en ajoutant 10H au contenu de BX le mot contenu dans CS (Ici WORD PTR est facultatif, car la précision du registre CS implique qu'il s'agit d'un mot).

MOV WORD PTR [BX+0E],0000 : Met en mémoire à l'adresse (sur DS:) calculée en rajoutant 0EH au contenu de BX le mot de valeur 0000.

MOV BYTE PTR [BX+0D],01 : Met en mémoire à l'adresse (Sur DS:) calculée en ajoutant 0DH au contenu de BX l'octet de valeur 01H.

ES :MOV BYTE PTR [0200],FC : Met en mémoire à l'adresse ES:0200 (ES: joue un rôle similaire à CS:) l'octet de valeur FCH.

POP DI : Dépile un mot et en met le contenu dans le registre DI.

PUSH AX : Empile le mot contenu dans AX.

REP MOVSB : Répètition (REP) de déplacement (MOV) de chaine (String) octet par octet (Byte). **MOVSB** recopie l'octet de l'adresse DS:SI à l'adresse ES:DI, et si l'indicateur de direction est à 0 (voir CLD) incrémente SI et DI.

REP décrémente le contenu de CX et répète cette opération tant que le contenu de CX est différent de 0000.

RETF : RETurn Far, fin de sous-programme dont l'appel a été fait par un CALL FAR, c'est-à-dire que l'adresse segment:déplacement a été utilisée.

SHL AX,1 : Décalage (SHift) sur la gauche (Left) du contenu de AX d'1 bit.

SHL DX,CL : Décalage sur la gauche du contenu de DX du nombre de bits contenus dans CL.

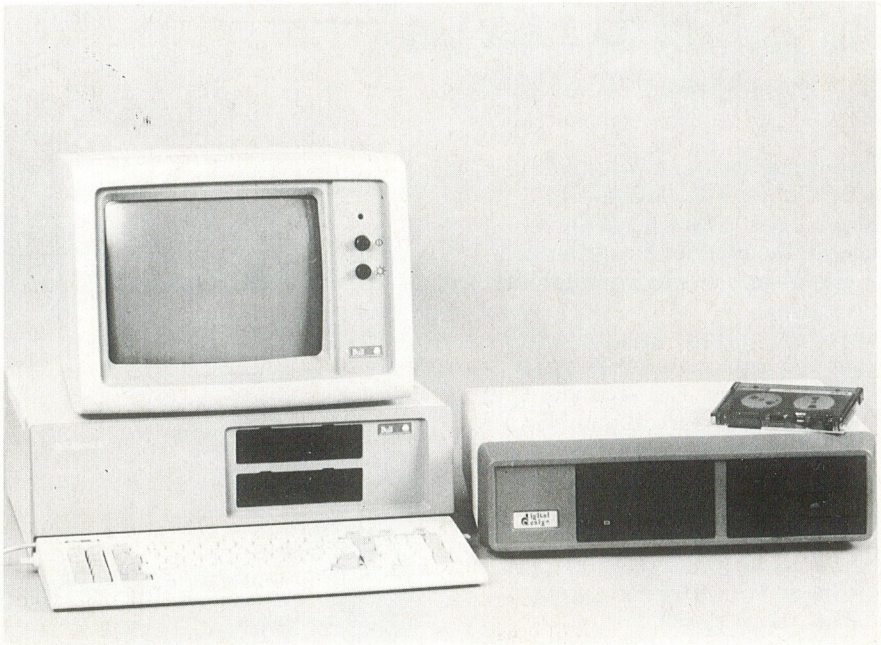
SUB CX,DI : Soustrait au contenu de CX le contenu de DI.

XOR AH,AH : Ou eXclusif du contenu de AH avec le contenu de AH, le résultat sera inévitablement nul.

XOR SI,SI : Ou eXclusif du contenu du registre SI avec le contenu du registre SI.

Des sous-ensembles

Digital Design se spécialise, de plus en plus, dans la fabrication et la commercialisation de sous-ensembles de mémoire de masse magnétique.
Ces sous-ensembles sont équipés d'une unité de disque dur 5''1/4 Winchester, 30 à 140 Méga octets à haute vitesse (30 ms de temps moyen d'accès) et d'un système de sauvegarde rapide sur cartouche à bande de capacité 60 Méga octets.
Ils sont compatibles sur les systèmes *Bull Micral 9050* et *9020* sous *Prologue !* et sur les systèmes *Bull Micral 30, Bull Micral 60, IBM PC-XT (ou compatibles) et IBM AT ou Compatibles* sous *Prologue* ou *MS DOS.*



Le premier buffer compatible tous micros-ordinateurs

Interquadram annonce *Microfazer II,* le premier *buffer* qui combine sur une seule carte tous les modes d'interfaçage micro/imprimante.

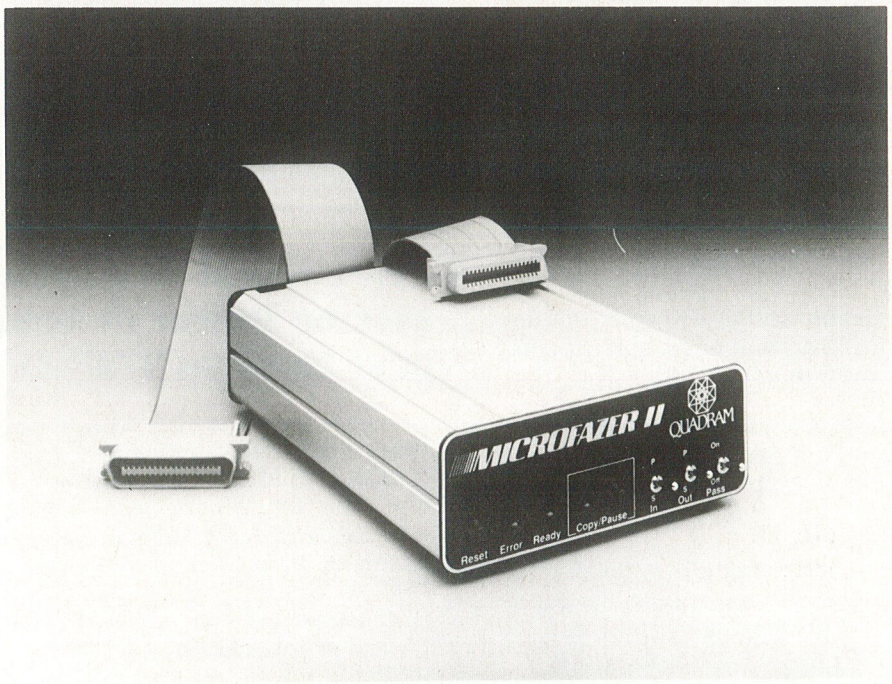
Doté d'une mémoire de 8 Ko extensible à 2 Mo, *Microfazer II* dispose en standard des modes entrée-sortie parallèle/parallèle, parallèle/série, série/parallèle, série/série. Il permet de connecter imprimantes, tables traçantes, et modems à n'importe quel micro-ordinateur équipé d'une sortie parallèle *Centronics* ou *série RS-232 C.*

Grâce à sa capacité mémoire, il est parfait pour les applications nécessitant jusqu'à 2 Mo de mémoire, telles que l'impression de tableurs, de graphiques ou le transfert de fichiers via un modem.

Ce nouveau *buffer* imprime les données sans utiliser la mémoire de l'ordinateur et libère le *PC* pendant l'impression. Il accepte l'insertion de données supplémentaires en cours même d'impression.

Différents modes de fonctionnement (pause, copy, bypass) sont accessibles à partir de la face avant de l'appareil.

Le *Microfazer II* est actuellement le *seul buffer sur le marché,* qui propose plus de 512 K de mémoire vive et qui combine tous les modes d'interfaçages sur une seule carte.
Le *Microfazer II* est dès ce mois de mai.

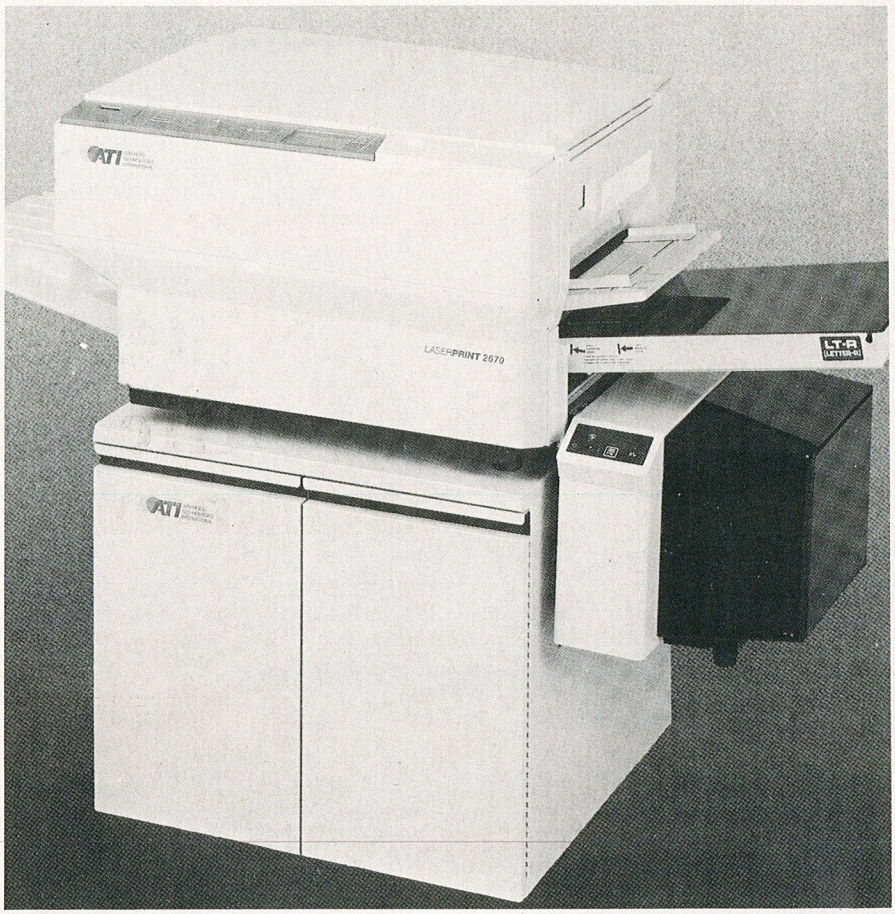


Une garantie de 3 ans pour une imprimante à laser

Avec l'imprimante à *laser 2670, ATI* pour un prix défiant toute concurrence, c'est une qualité parfaite de courrier en instantané de graphisme, et une comptabilité universelle.
Elle peut imprimer jusqu'à 1 500 lignes à la minute, soit 26/pages minutes, 1 page/2, 3 secondes.
2 fois plus vite que de nombreuses imprimantes en ligne, 3 fois plus rapide que d'autres lasers, 20 fois plus rapide que les imprimantes à marguerite.
300 points au pouce (118 pts au cm), pour vous étonner par sa qualité d'impression.
Silencieuse, elle n'est d'aucune gêne pour tout environnement de travail.
Une seule touche sur le clavier et vous avez le choix entre une grande variété d'impressions et de polices.
Toutes les possibilités de graphisme sont offertes.
Grâce à sa conception modulaire, la *2670* est compatible avec presque tous les systèmes, en particulier avec *IBM*. Une des plus souples d'emploi de sa génération, elle est pourtant nettement moins chère que toute autre imprimante qui serait très loin d'avoir tous ses talents.
Sa conception est une merveille de simplicité. Elle concilie l'art et la technique éprouvée.
C'est le mariage d'un scanner laser semi-conducteur à la technique xérographique fiable et d'un moteur d'impression contrôlée par le microprocesseur *68000,* standard de l'industrie.

Les innovations se signalent dans le convertisseur de protocole universel, les contrôleurs d'interfaces modulaires enfichables par carte et les options intelligentes de manipulation de papier, idéales pour la gestion et les applications scientifiques.

Silencieuse et rapide, la *2670* fait sans problème le travail de plusieurs imprimantes à aiguilles, en ligne, traceur de courbes et marguerite.
Actuellement plusieurs polices peuvent être mixées sur une page.
On peut introduire le logo une compagnie ou d'une signature.
La *2670* plus le système bit-map, est idéale pour de nombreuses applications scientifiques d'engineering et de gestion. Il est facile de générer des impressions, des graphismes, des histogrammes, des tracés et des dessins en 3 dimensions.
Les graphiques peuvent être accompagnés de textes dans le même document.
ATI 2670 est garantie par *3M France* sur site pièces et main-d'œuvre pendant 3 ans, les seuls coûts variables de cette imprimante n'étant que les consommables. Et l'on arrive donc à un coût à la page d'environ 8 centimes.



Pérex élargit sa gamme Péristream avec un lecteur de bande en continu pour cassettes 1/4'' au format SCSI

Perex Ltd, premier fabricant européen de lecteurs de cassettes 1/4'' pour systèmes de saisie et de sauvegarde, complète aujourd'hui sa gamme *Peristream de streamers 1/4'' mi-hauteur pour disques Winchester* haute capacité en lançant une version au format *SCSI.*

Ce nouveau modèle est capable de sauvegarder 45 Mo en moins de 12 minutes et offre une capacité de stockage après formatage de 45 Mo sur cassette standard *DC300XL,* ou de 60 Mo sur bande haute coercivité *DC600A.*

POUR TROUVER TOUTES LES INFORMATIONS SUR LES ENTREPRISES, LES PRODUITS ET SUR L'ACTUALITE DE LA MICRO-INFORMATIQUE PROFESSIONNELLE.

Profitez de notre offre spéciale d'abonnement à

OFFRE SPECIALE

Bulletin d'abonnement

à adresser à :
SOFTIN COMMUNICATION
2, rue d'Amsterdam
75009 Paris

M. .....

.....

Adresse .....

.....

souhaite bénéficier de l'offre spéciale d'abonnement à IBM-PC et COMPATIBLES HEBDO au prix spécial de 350 F TTC au lieu de 395 F TTC

☐ chèque ci-joint

☐ à facturer

CAF : MISSION ACCOMPLIE
SUCCÈS CAF : SÉRIE PC. MASTER
L'ÉVOLUTION VERS L'UNIVERS PC/AT

PC. MASTER. 401 CONFIGURATION DE BASE

- Micro-processeur : 80286 - 6 MHZ - Timer 10 MHZ - 8 slots .
- Mémoire : 512 K extensible directement à 1024 K RAM sur carte mère et à 16000 K par adjonction de cartes.
- Contrôleur deux disquettes et disque dur.
- Une unité de disquette 5" 1/4 1200 K.
- Carte graphique couleur (sorties monochrome et couleur).
- Deux interfaces série RS. 232 et un interface parallèle imprimante.
- Clavier français AZERTY.
- Alimentation 195 W.
- DOS* 3.1.
- OPTIONS DOS : UNIX* (Venix Système 5) XENIX*. MOS*. PROLOGUE*.

PC. MASTER.402 Même configuration que PC. MASTER. 401 excepté deux unités de disquettes au lieu d'une .

PC. MASTER UN COMPATIBLE IBM/AT QUI CUMULE LES PLUS.

PC. MASTER LA MÉMOIRE D'UN MINI : 512 K à 16000 K RAM.

PC. MASTER UN CHOIX DE DISQUES : 20 Mo à 70 Mo.

PC. MASTER LA SÉCURITÉ EN PLUS GRÂCE AU STREAMER INTÉGRÉ AVEC DES SYSTÈMES D'EXPLOITATION mono et multitaches DOS 3.1 - UNIX, XENIX Prologue - MOS. ET LE SERVICE OMNIUM-PROMOTION : 12 ANNÉES D'EXPÉRIENCE INFORMATIQUE.

PC MASTER : DES PRIX « CHOCS ». PC MASTER toujours mieux et tellement moins cher

*Marques déposées - DOS : Microsoft - UNIX : Bell Laboratories - XENIX : Microsoft MOS : Interlogiciel - Prologue - IBM : International Business Machines Corporation.

PC. MASTER 403. 512 K - 1 disquette 1200 K - 1 disque 20 Mo
Même configuration que PC. MASTER 401 avec en plus un disque dur 20 Mo.

PC. MASTER 404. 512 K - 2 disquettes 1200 K - 1 disque 20 Mo
Même configuration que PC. MASTER 402 avec en plus un disque dur de 20 Mo.

PC. MASTER 443. 512 K - 1 disquette 1200 K - 1 disque 40 Mo.
Même configuration que PC. MASTER 401 avec en plus un disque dur de 40 Mo.

PC. MASTER 483 : 1024 K - 1 disquette 1200 K - 1 disque dur 80 Mo.

OPTIONS :

Streamer intégré de 20 Mo - 60 Mo.
Carte EGA : Haute résolution monochrome et couleur (640 x 350) - 7 modes de résolution écran graphique.

RECHERCHONS REVENDEURS

Bulletin-réponse :

Société : _____

_____

Adresse : _____

_____

Nom du responsable : _____

Tél. : _____

IMPORTATEUR EXCLUSIF EN FRANCE

OMNIUM PROMOTION

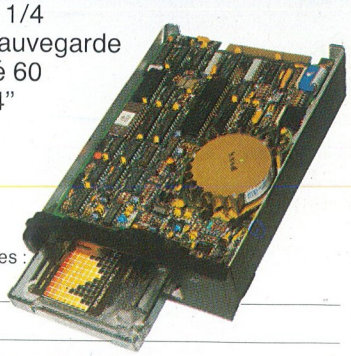
110, avenue Marceau, 92400 COURBEVOIE
Tél. 47.88.51.42 - Télex PROMIUM 610.136 F

Services Techniques : 11, rue de Cambrai, 75019 PARIS
Tél. : 42.02.17.73 - Télex PROMIUM 213.084 F

LES PÉRIPHÉRIQUES QUI FONT L'INFORMATIQUE

ENTREPRISE ET COMMUNICATION — PHOTO PICTOR INTERNATIONAL

Le STREAMER KENNEDY 5" 1/4 est la solution idéale pour la sauvegarde de vos disques durs. Capacité 60 méga-octets en cartouche 1/4" 3M standard série 600.
Le STREAMER KENNEDY est disponible en interface QIC 36 - QIC 02 - SCSI.



Pour tous renseignements complémentaires :

Mr, Mme : _____

Société : _____

Adresse : _____

Souhaite recevoir une documentation sur les Streamers Kennedy.

les périphériques qui font l'informatique

DRI

Newbury Data

Membre du groupe D.R.I.

Téléphone : 39.56.81.11

655, avenue Roland Garros - 78530 BUC-FRANCE

U.IBM/PC

SINTELEX 103D

le système complet qui apporte la communication télex à votre IBM-PC ou compatibles

Le nouveau standard TELEX pour IBM-PC et compatibles.

CARACTÉRISTIQUES

- Saisie des messages télex sur votre traitement de textes *habituel* (Text, Wordstar, etc.).
- Messages télex : fichiers MS-DOS. Intégration des messages dans vos bases de données...
- Tri des messages télex. Exemple : retrouver tous les messages télex envoyés à un destinataire donné ou dans une période donnée.
- Suivi de toutes les transactions grâce au Journal de Service. (Récapitulatif des envois et réceptions).
- Mémorisation ou impression des messages dans le Sintelex si l'ordinateur est occupé ou éteint.
- Fonctionnement SILENCIEUX.

OPTIONS

- Mode conversationnel.
- Envoi différé.
- Envoi d'un groupe de messages.
- Intégration dans un réseau local.
- Connection à des systèmes serveurs Minitel.

PROPRIÉTÉS TECHNIQUES

1. Se relie au PC par ligne. Série RS 232 C.
2. Transmission PC-Sintelex à 2.400 bauds.
3. Transmission sur ligne Télex 50 bauds.
4. Dimensions : 8 cm × 39 cm × 27 cm. Poids : 2 kg.

EVROZ électronique

40, rue d'Hautpoul
75019 PARIS
Tél. : 203.30.43 - Télex 216517

Importateur exclusif
Recherchons revendeurs

AGRÉÉ PTT N° 85018 T du 20.3.85. Catégorie 1.

MAINTENANCE ASSURÉE.